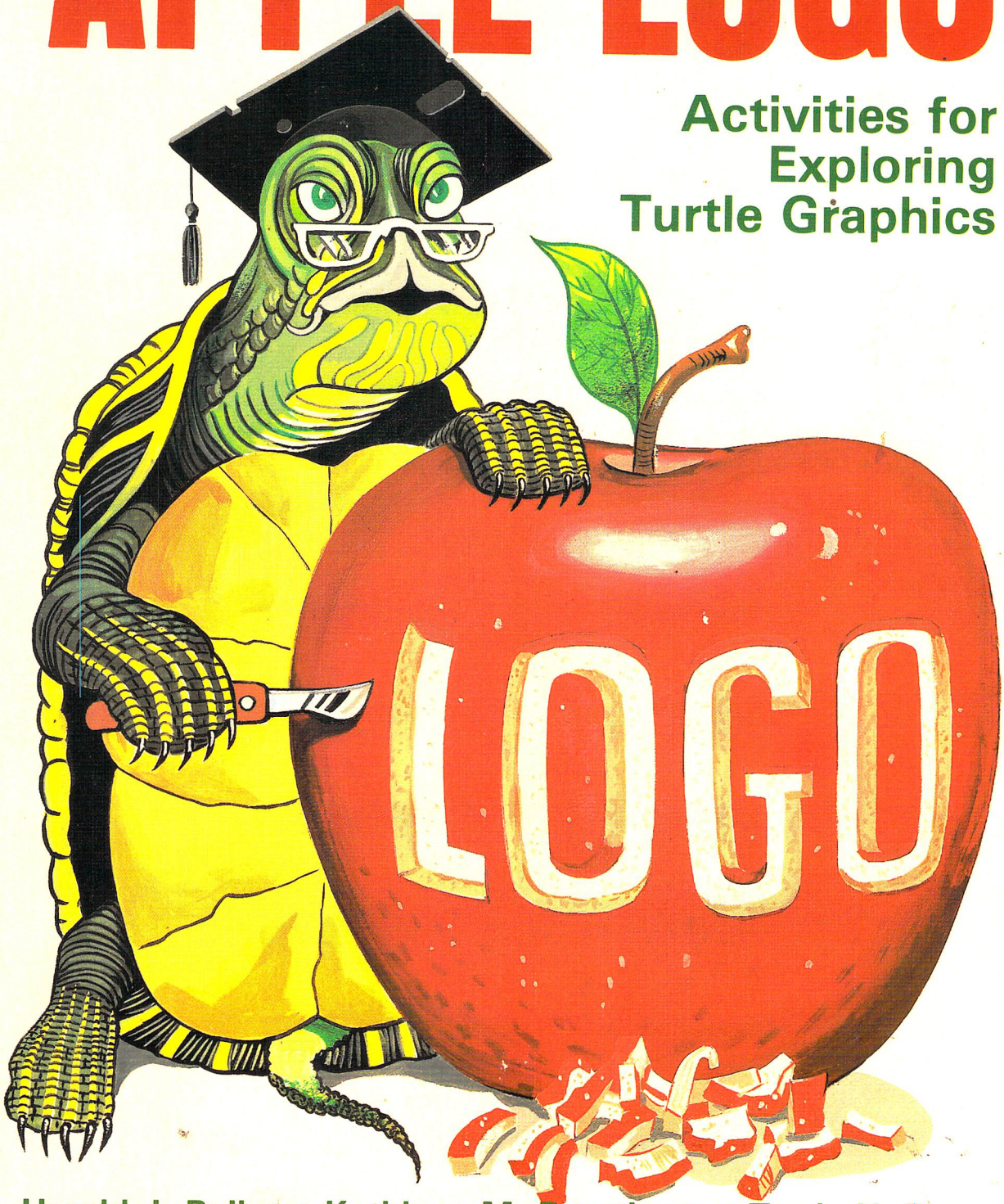
APPLE LOGO

Activities for Exploring Turtle Graphics



Harold J. Bailey • Kathleen M. Brautigam • Trudy H. Doran

Apple Logo

Publishing Director: David Culverwell
Acquisitions Editor: Gisele Asher
Production Editor/Text Design: Michael J. Rogers
Art Director/Cover Design: Don Sellers
Assistant Art Director: Bernard Vervin
Photographer: Leslie McMahon
Manufacturing Director: John Komsa

Typesetting: Creative Communications Corporation, Baltimore, MD
Printing: Fairfield Graphics, Fairfield, PA
Copy Editor: Carol Thorsten-Stein
Typefaces: Eurostile (display heads), Palatino (text), and Typewriter (Logo programs)

Apple Logo

Activities for Exploring Turtle Graphics

Harold J. Bailey
Bloomsburg University
Bloomsburg, Pennsylvania

Kathleen M. Brautigam
Milton Area School District
Milton, Pennsylvania

Trudy H. Doran
Shikellamy School District
Sunbury, Pennsylvania

Brady Communications Company, Inc.
A Prentice-Hall Publishing Company
Bowie, MD

Apple Logo: Activities For Exploring Turtle Graphics

Copyright © 1984 by Brady Communications Company, Inc.
All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage and retrieval system, without permission in writing from the publisher. For information, address Brady Communications Company, Inc., Bowie, Maryland 20715.

Library of Congress Cataloging in Publication Data

Bailey, Harold J., 1940–
Apple LOGO
Includes index.
1. Computer graphics. 2. LOGO (Computer program)—Language. 3. Apple II Plus (Computer)—Programming. 4. Apple IIe (Computer)—Programming. 5. Franklin Ace 1000 (Computer)—Programming. I. Brautigam, Kathleen M. II. Doran, Trudy H. III. Title. IV. Title: Turtle graphics.
T385.B344 1984 001.64'43 83-26606

ISBN 0-89303-312-X

Prentice-Hall International, Inc., London
Prentice-Hall Canada, Inc., Scarborough, Ontario
Prentice-Hall of Australia, Pty., Ltd., Sydney
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Prentice-Hall of Southeast Asia Pte. Ltd., Singapore
Whitehall Books, Limited, Petone, New Zealand
Editora Prentice-Hall Do Brasil LTDA., Rio de Janeiro

Printed in the United States of America

84 85 86 87 88 89 90 91 92 93 94 1 2 3 4 5 6 7 8 9 10

Dedications

To Brad, for being my best friend.

Kathy

To my parents, for years of support, love, and encouragement.

Trudy

To my family, for their continual patience and understanding during a period when my professional aspirations have affected our moments together.

Hank

Contents

Preface / ix

Chapter 1 Exploring Turtle Territories / 1
Activity 1.1 Turtle Moves / **1**
Activity 1.2 Turtle Pivots / **6**
Activity 1.3 Turtle Toggles / **12**
Activity 1.4 Turtle Repetitions / **17**
Activity 1.5 Turtle Circles / **22**
Activity 1.6 Turtle Arcs / **28**
Reference Point One / **32**

Chapter 2 Preserving Turtle Tracks / 35
Activity 2.1 Turtle Procedure / **35**
Activity 2.2 Turtle Files / **43**
Activity 2.3 Turtle Inventory / **50**
Activity 2.4 Turtle Editing / **55**
Activity 2.5 Turtle Controls / **66**
Activity 2.6 Turtle Maintenance / **72**
Reference Point Two / **82**

Chapter 3 Extending Turtle Trips / 85
Activity 3.1 Turtle Input / **85**
Activity 3.2 Turtle Hues / **95**
Activity 3.3 Turtle Background / **100**
Activity 3.4 Turtle Variables / **109**
Activity 3.5 Turtle Iterations / **119**
Activity 3.6 Turtle Limits / **127**
Activity 3.7 Turtle Conditions / **140**
Activity 3.8 Turtle Tricks / **152**
Reference Point Three / **158**

Chapter 4 Coordinating Turtle Points / 167
Activity 4.1 Turtle Positions / **167**
Activity 4.2 Turtle Dots / **176**
Activity 4.3 Turtle Placement / **188**
Activity 4.4 Turtle Tessellations / **194**
Reference Point Four / **201**

Chapter 5 Creating Turtle Temptations / 209
Introduction / **209**
Part-To-Whole Programming / **210**
Whole-To-Part Programming / **216**

Concluding Remarks / 223
Suggested Projects / 224

Resource List / 225

Answers To Selected Turtle Testers / 229

Index / 241

Preface

Who enjoys drawing more than children? Why are art classes usually included in the public school curriculum? It is because youngsters find the drawing of pictures an enjoyable and effective way of expressing their ideas and thoughts.

Traditionally, children have used pencil, crayon, and paint to demonstrate their artistic talents. The microcomputer makes a new world of graphics and color available to everyone.

One of the computer languages with graphics capabilities is Logo. Logo represents a philosophy of education advocated by Seymour Papert at the Massachusetts Institute of Technology. Through Logo, Papert maintains that children gain control over the learning process by teaching the computer, rather than being taught by it. Logo provides an environment where the computer becomes a tool to facilitate learning. A child first explores and plays with an idea and observes the results immediately through the path of a "turtle" illustrated in the picture below.

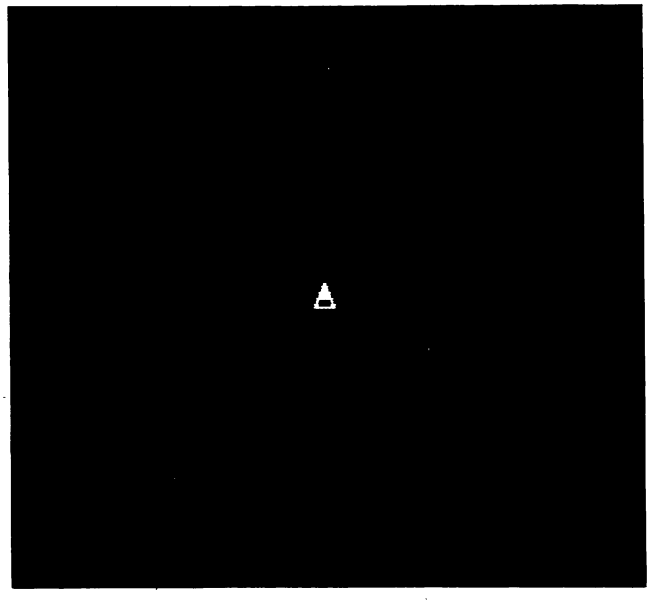


Figure P-1.

The youngster discovers relationships among several ideas and formalizes them by creating a procedure. The development of procedures becomes a personal repertoire of building blocks through which logical thinking, problem solving skills, and creativity are enriched.

There are versions of Logo for virtually every popular home computer. Several are designed specifically for the Apple system. The purpose of this

book is to demonstrate how the Apple Logo version uses Turtle Graphics to facilitate the application of Papert's theory. While Apple Logo also has arithmetic operations and list-processing capabilities, this text concentrates solely on Turtle Graphics.

To use this book, you must have an Apple II Plus, an Apple IIe, or a Franklin Ace 1000 computer system with at least 48K of memory. The Apple II Plus system requires an additional 16K byte memory expansion (such as the Apple Language Card). You also need at least one disk drive to use your Apple Logo diskette. Access to a color monitor or a color TV is desirable to fully appreciate the program examples in the text.

This book consists of a collection of instructional activities that progress in a systematic manner, culminating in a comprehensive understanding of Turtle Graphics. No previous computer experience is required. All Logo commands, geometric concepts, and elementary computer jargon are defined when introduced, reinforced throughout the book, and summarized at the end of each chapter.

The selection and sequence of activities within each chapter are based upon the authors' teaching experience and interaction with children exploring Logo. Integrating this practical knowledge of an appropriate learning sequence, *Apple Logo: Activities For Exploring Turtle Graphics* provides a more direct and accelerated approach for the following audiences.

Parents: offers an opportunity for parents and children to explore Logo together

Educators: provides an instructional guide for introducing Logo to children

Prospective educators: serves as a textbook for a college/university level Logo course

Independent learners: allows for individualized study of Logo for the able student and motivated hobbyist aged 12 and up.

Each activity includes five sections. They are:

Turtle Trials
Turtle Talk
Turtle Truth
Turtle Tester
Turtle Teaser

The **Turtle Trials** section presents a series of "hands-on" activities where the reader sits at a microcomputer and enters given Logo commands. The reader is asked to describe what is observed in the resulting displays. These results demonstrate both the proper usage and common errors associated with the activity objective. You are encouraged to write descriptions, notes, and observations in the book.

The **Turtle Talk** section provides narrative reinforcement for the interactive "hands-on" experience.

The **Turtle Truth** section provides a comprehensive example that demonstrates a correct application of the newly introduced commands and presents one of the many possible graphic designs using Turtle Graphics.

The **Turtle Tester** section requests the reader to demonstrate an understanding of the activity objective by writing a Logo program for a given situation.

Finally, the **Turtle Teaser** section suggests an activity of an extended nature to further explore, refine, and integrate Logo skills.

The Logo activities form five chapters which should be followed sequentially. Chapter 1 shows how one controls the movement of the "turtle" through simple Logo commands. In Chapter 2, these commands and others are combined to form procedures. The reader also becomes skillful in using the Logo editor. Chapter 3 introduces color and the concept of recursion, to extend the development of procedures to include variables. In Chapter 4, coordinates are used as a means by which graphic designs can be displayed and animated in an effective manner. Finally, Chapter 5 presents several different approaches by which complex screen displays can be designed.

Logo is designed to provide a medium for meaningful and enjoyable learning experiences. We believe that *Apple Logo: Activities For Exploring Turtle Graphics* will guide the reader toward a proficiency in the use of Turtle Graphics and provide an opportunity for creative expression.

H.J.B.
K.M.B.
T.H.D.

Limits of Liability and Disclaimer of Warranty

The authors and publisher of this book have used their best efforts in preparing this book and programs contained in it. These efforts include the development, research, and testing of the programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs, the text, or the documentation contained in this book. The authors and publisher shall not be liable in any event for claims of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the text or the programs. The programs contained in this book are intended for the use of the original purchaser.

Note to Authors

Do you have a manuscript or a software program related to personal computers? Do you have an idea for developing such a project? If so, we would like to hear from you. The Brady Company produces a complete range of books and applications software for the personal computer market. We invite you to write to David Culverwell, Publishing Director, Brady Communications Company, Inc., Bowie, MD 20715.

Trademarks of Material Mentioned in This Text

Apple II Plus and Apple IIe are registered trademarks of Apple Computer, Inc.

Apple ® Logo is a trademark of Logo Computer Systems, Inc.

Franklin Ace 1000 is a registered trademark of Franklin Computer Corporation.

Acknowledgments

The authors gratefully acknowledge the following students and staff who inspired the development of the materials in *Apple Logo: Activities For Exploring Turtle Graphics.*

Bloomsburg University Computer Camp
Milton Area School District
Shikellamy School District

Special thanks are extended to the following colleagues and associates for their help in reviewing, testing, and critiquing our manuscript.

Charlotte Albertine
Jane Kaar
Tim Shannon

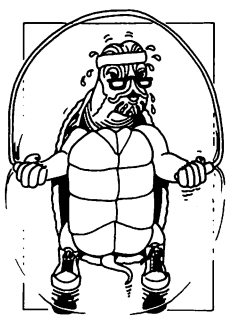
Finally, we express our gratitude to each family and many friends for their patience and constant support during the sometimes hectic period of writing this book.

Brad, Kristin, and Phillip
David, Darci, and Tonya
Linda, Kimberly, and Keith
and many encouraging friends

1

Exploring Turtle Territories

Activity 1.1 Turtle Moves



Turtle Trials. This activity introduces fundamental commands that control the appearance and movement of the symbolic "turtle" used in Logo.

1. Open the disk drive door, insert the Apple Logo diskette, close the disk drive door, then turn the computer and monitor (or TV) on. This procedure is known as "cold booting" the computer. You will receive the following message:

```
PRESS THE RETURN KEY TO BEGIN

IF YOU HAVE YOUR OWN FILE DISKETTE,
INSERT IT NOW, THEN PRESS RETURN

(C) LOGO COMPUTER SYSTEMS INC. 1982
```

Ignore the second set of instructions, as we have not created our own file diskette. This will be covered in Chapter 2. For now press RETURN without inserting another diskette. The disk drive will activate to transfer or load a group of Logo programs or procedures referred to as the STARTUP file. When the drive stops, you will receive the message

```
WELCOME TO LOGO                          A1.5
?■
```

The question mark is known as the PROMPT and indicates that the computer is ready to accept the instructions you type on the keyboard. These instructions or commands are commonly known as "input." The flashing box is called the CURSOR and shows exactly where the input will appear on the screen.

NOTE Please follow the loading procedure described above for all of the activities in Chapter 1. It will be assumed that you have completed this "booting" process prior to each activity.

Type the word HELLO and press the key marked RETURN. Describe the computer's response or screen display on the line provided.

HELLO _____

NOTE In all of the following activities you will be presented with directions and then a list of commands to type. Please assume that after typing the command you should press the RETURN key and observe the screen display. If a typing error occurs, it can be corrected by backspacing with the "left arrow" key (←) until the cursor erases the error and other characters it passes over. Retype the remainder of the line and press the RETURN key.

2. Enter the following sequence of commands and describe what effect each command has on the screen display.

CLEARSCREEN _____
FORWARD 40 _____
FORWARD 10 _____
FORWARD20 _____
FD 30 _____
FD _____
CS _____
HIDETURTLE _____
BACK 25 _____
SHOW TURTLE _____
SHOWTURTLE _____
BK 10 _____
HT _____
BK20 _____
BK 15 _____
ST _____
BK 5 _____
BK _____
CS _____

3. Enter the following commands and observe the screen display.

```
FD 40
FD 40
FD 30
FD 5
FD 5
FD 5
FD 40
```

Where is the turtle (wedge-like object) originally located on the screen?

_____

Starting from that original position, what is the approximate length that the turtle can move forward in "turtle units" before disappearing from the screen? _____

Describe where the turtle reappears. _____

4. Enter the following sequence of commands and observe the screen display.

```
CS
BK 30
BK 30
BK 20
BK 5
BK 60
```

Where is the turtle originally located on the screen? _____

Starting from that original position, what is the approximate length the turtle can move back in "turtle units" before disappearing from the existing display screen? _____

Describe where the turtle reappears. _____

5. Enter the following commands. Observe and describe the effect on the screen display.

```
CS
FD 135 _____
FULLSCREEN _____
TEXTSCREEN _____
SPLITSCREEN _____
```

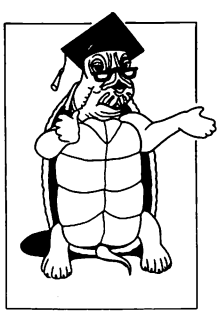
6. There is a key marked CTRL (CONTROL on the Apple IIe) on the keyboard. It is used as an alternative shift key to type what is called a "control character." For example, to type CTRL-S, you must hold down the CTRL key while you press the S key. To experiment, enter the following commands.

```
CS
FD 12Ø
FD 12Ø
CTRL-L _____
CTRL-T _____
CTRL-S _____
```

What is the vertical length of the screen when fullscreen display is in effect? _____
What full word command is equivalent to CTRL-L? _____
What full word command is equivalent to CTRL-T? _____
What full word command is equivalent to CTRL-S? _____

7. Enter the following sequence of commands and observe the screen display. Describe what happens after each section.

```
CS
ST
FD 6Ø
FD -6Ø _____
FD 7Ø
BK 7Ø _____
BK 4Ø
BK -4Ø _____
```



Turtle Talk. Part 1 demonstrates that access to Turtle Graphics is easily obtained by booting with the Logo diskette. We will insert and use a personal diskette in later activities when the saving and retrieving of Logo files becomes important. For now, it is unnecessary to save any information on this diskette.

In Part 2, CLEARSCREEN causes a small wedge-like object known as the turtle to appear on the screen. This turtle simulates the communication between you and drawings produced on the display screen. The appearance and disappearance of the turtle is controlled by the commands SHOWTURTLE (abbreviated ST) and HIDETURTLE (abbreviated HT). If a space is inserted between the words of these commands, an error message is given. CS is an abbreviation for CLEARSCREEN. It is used to erase all drawings and place the turtle in the center of the screen pointing up.

FORWARD (abbreviated FD) and BACK (abbreviated BK) are movement commands introduced in Parts 3 and 4. FD moves the turtle in the direction in which it is pointing. BK moves the turtle in the opposite direction. The length of the path taken by the turtle in "turtle units" is defined by

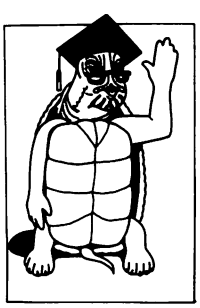
the number following the FD or BK command. If no number follows the command, an error message such as NOT ENOUGH INPUTS TO FORWARD occurs. The command must then be re-entered properly.

Part 7 demonstrates that it is possible to enter a negative value as input following an FD or BK command. This causes the turtle to move in the reverse direction of the command used. The incorporation of negative values is important when demonstrating certain mathematical principles. Note that while negative inputs are permitted, only positive values are needed since both the FD and BK commands are available for providing turtle movement in opposite directions.

Part 5 and 6 show there are three methods of viewing the screen. SPLITSCREEN (CTRL-S) is the normal way of viewing the screen. This allows you to view graphics and also see four lines of text. The screen dimensions in this display mode are 120 units forward (FD) from the turtle's starting position and 85 units back (BK) from the turtle's starting position.

FULLSCREEN (CTRL-L) lets you see the total graphics screen, which is 240 units high or 120 units forward (FD) from the turtle's starting position and 120 units back (BK) from the turtle's starting position. This display temporarily removes the four lines of text which appear at the bottom of the screen, commonly called the "text window." This lets you see if the turtle is drawing in the section under the text window. If a command is given that sends the turtle off the screen, it wraps around to reappear and continue drawing at the opposite edge of the screen.

TEXTSCREEN (CTRL-T) lets you view the commands that have been previously entered, but have scrolled off from view in the text window. The display screen commands do not erase the existing drawing or text.



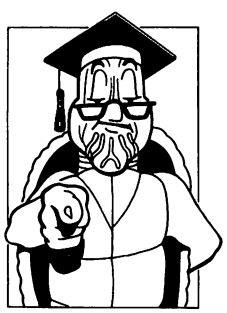
Turtle Truth. Study the following commands to determine the screen display. Check your answers by entering the commands.

```
CS
BK 5Ø
HT
FD 8Ø
ST
FD 4Ø
FD 1ØØ
CTRL-L
```

```
CS
CTRL-T
CTRL-S
HT
FD 4Ø
FD -1ØØ
CS
ST
```

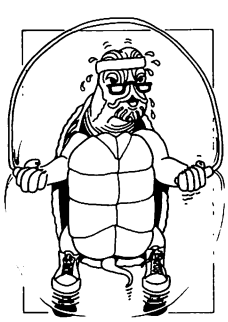
Turtle Tester. Write a sequence of commands which performs the following operations in this order.

A. Clear the screen and then hide the turtle.
B. Move forward 120 units.
C. Clear the screen, then show the turtle.
D. Move back 75 units. (Answer on page 229.)



Turtle Teaser. Use no more than four Logo commands to draw a vertical line from the top of the screen to the bottom of the screen in FULLSCREEN mode.

Activity 1.2 Turtle Pivots



Turtle Trials. This activity introduces the use of commands to set the heading of the turtle.

NOTE HEADING is the term used to denote the direction of the turtle. When the turtle first appears on the screen, the starting position of the tip will always be northerly.

1. Enter the following sequence of commands and notice the change in HEADING shown by the turtle. Describe what happens after each command.

```
CS _____
RT 90 _____
FD 25 _____
CS _____
RT _____
RT 180 _____
FD 50 _____
CS _____
RT 270 _____
FD 75 _____
CS _____
RT360 _____
RT 360 _____
FD 100 _____
```

2. Enter the following sequence of commands and notice the change in HEADING shown by the turtle. Describe what happens after each command.

```
CS _____
LT 90 _____
FD 50 _____
BK 50 _____
LT 90 _____
FD 50 _____
BK 50 _____
LT 90 _____
FD 50 _____
BK 50 _____
LT 90 _____
FD 50 _____
BK 50 _____
```

NOTE An angle is a figure formed by two lines extending from a common point. The size or measure of an angle can be expressed in degrees, where a degree represents a 1/360 part of the circumference (or perimeter) of a circle. Figure 1-1 illustrates a 45 degree angle. The LT and RT commands change the direction of the turtle, permitting it to revolve in an angular fashion.

How many degrees does the turtle revolve with LT 90? _____
How many degrees does the turtle revolve with RT 270? _____
How many degrees does the turtle revolve when it completes one revolution by returning to its starting position? _____

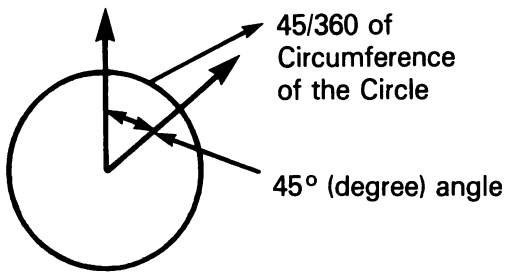


Figure 1-1.

3. Enter the following commands and observe the heading of the turtle after each section.

```
CS
RT 45
LT 45
```

Describe what happens. _____

```
RT 180
RT 180
```

Describe what happens. _____

```
LT 90
LT 180
```

Describe what happens. _____

```
LT 15
LT 50
LT 25
```

Describe what happens. _____
What is the sum of the last three input numbers? _____
Does the turtle draw when using an LT or RT command? _____
What does it do? _____
What commands must you use to make it draw? _____

4. Figures 1-2 and 1-3 display the turtle in its starting position. Imagine the turtle making RIGHT and LEFT turns. Fill in the numerical values that indicate the missing degrees of revolution for each of the lettered positions.

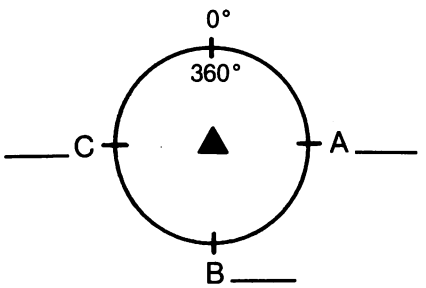


Figure 1-2.

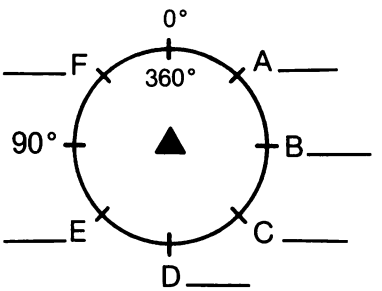


Figure 1-3.

5. Enter the following commands. Describe what happens after each command.

NOTE SETH represents SETHeading.

```
CS
SETH 90 _____
FD 100 _____
SETH 0 _____
FD 30 _____
SETH 270 _____
FD 100 _____
SETH 180 _____
FD 30 _____
SETH 360 _____
```

Describe the shape that is drawn. _____
Are the SETH commands based on RT or LT commands? _____
Why is SETH 360 the last command? _____
What is another command that could have been used instead of SETH 360? _____

6. Enter the following commands.

NOTE PR represents PRint.

```
CS
RT 97
LT 42
RT 405
PR HEADING
```

What reply does the computer give you to the command PR HEADING? _____
What two commands can you type to return the turtle to the original

HOME position from its current heading? RT _____
or LT _____

7. Enter these commands again, but add the new HOME command at the end.

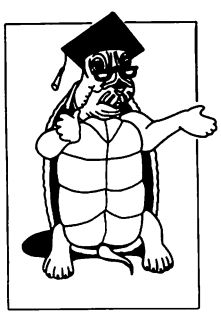
```
CS
RT 97
LT 42
RT 405
PR HEADING
HOME
```

What other commands can be used to return the turtle to its starting position? LT _____ or RT _____

8. Enter the following commands and observe the screen display.

```
CS
RT 105
FD 40
LT 37
FD 90
HOME
```

Describe the position of the turtle. _____
What geometric figure is drawn? _____

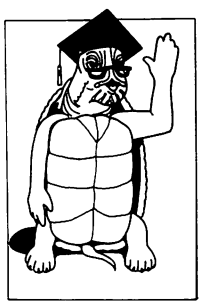


Turtle Talk. The turtle is represented by a triangular object with a solid tip. The direction this tip is facing is called the HEADING. The commands RT and LT, introduced in Parts 1, 2, and 3, use an input value to determine the number of degrees the turtle turns. The RT command provides clockwise revolution while the LT command provides counterclockwise revolution. If the commands RT or LT are entered without an input value, the error message "NOT ENOUGH INPUTS TO RIGHT" (OR LEFT) will appear. Notice that the space remains an important part of the command. Changing the heading of the turtle does not make it draw. Only FD or BK followed by a number allows the turtle to draw. By using these four commands in sequence, you are able to draw in any direction. A complete revolution of the turtle is 360 degrees. It will turn right or left from any starting point.

Part 5 shows that you can use the SETH (SETHeading) command with an input number to point the turtle to a predefined direction. The SETH command heading is determined by the clockwise revolution of the turtle in relation to its HOME position. The commands HOME, SETH 0, and SETH 360 appear to have similar control over the heading of the turtle.

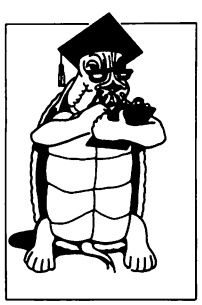
Each command turns the turtle in a northern direction. However, Part 8 demonstrates the HOME command will also send the turtle to the center of the screen and draw a path along the way.

If you get confused concerning the heading of the turtle, you can ask the computer to print the present heading. The command PR HEADING in Parts 6 and 7 causes the computer to print the heading on the screen. The PR HEADING output gives the numerical value of the current clockwise direction relative to the HOME position. In Part 5 the Logo commands were terminated with SETH 360 or HOME. It is good practice to return the turtle to its starting (HOME) position after each drawing.



Turtle Truth. Study the following Logo commands to determine the screen display. Check your answer by entering the commands.

```
CS
ST
RT 90
FD 100
LT 90
FD 50
SETH 270
FD 200
LT 90
FD 50
SETH 90
FD 200
BK 100
HOME
PR HEADING
```



Turtle Tester. Draw three rectangles of different sizes. Return the turtle to its starting position after the drawing is completed.

A. Draw a rectangle using only RT commands for the corners. Make the rectangle tall. (Answer on page 229.)
B. Draw a rectangle using only LT commands for the corners. Make the rectangle wide. (Answer on page 229.)
C. Draw a rectangle any way you want, but use only SETH commands for the corners.

Turtle Teaser. Draw the rectangle illustrated in Figure 1-4.

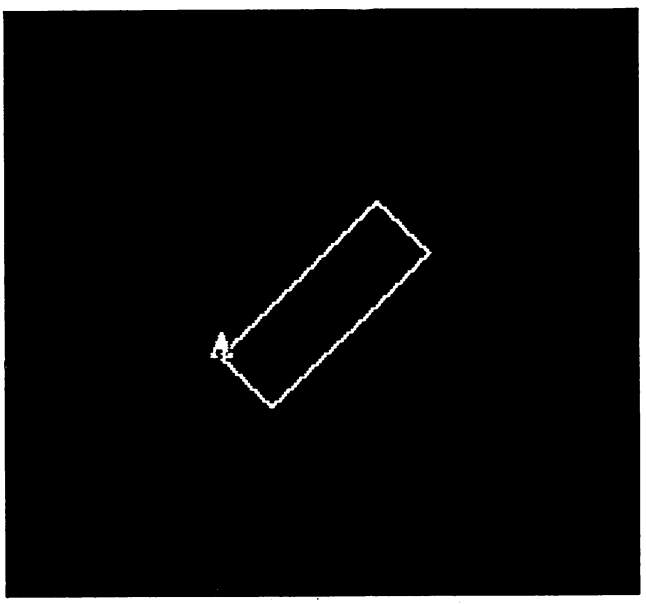
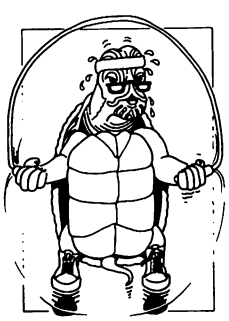


Figure 1-4.

Activity 1.3 Turtle Toggles



Turtle Trials. This activity introduces the commands that permit the user to relocate the turtle and alter its drawing mode.

1. Enter the following commands.

```
CS
LT 18Ø
PENUP
FD 4Ø
```

What kinds of movement do you observe? _____

2. Add the following commands and describe the effect each command has on the screen display.

```
RT 9Ø _____
FD 4Ø _____
RT 9Ø _____
```

Does any drawing take place? _____

3. Add the following commands.

```
PENDOWN
FD 8Ø
```

What do you observe? _____

4. Add the following commands.

```
RT 9Ø
FD 4Ø
RT 9Ø
FD 8Ø
RT 9Ø
FD 4Ø
```

What geometric shape is drawn? _____
What is the heading? _____
Type the command

```
PR HEADING
```

What is the computer's response? _____

5. Type the following commands and describe the effect each command has on the screen display.

```
PENERASE _____
BK 4Ø _____
FD 4Ø _____
BK 4Ø _____
```

6. Add the following commands.

```
PD
FD 4Ø
```

```
PE
BK 4Ø
PD
```

What command places the turtle in the erase mode? _____

```
CLEAN
```

Describe what happens. _____

7. Enter the commands and observe what takes place after each section.

```
HOME
CS
LT 9Ø
FD 5Ø
RT 9Ø
FD 8Ø
BK 16Ø
CS
```

Describe what happens. _____

```
LT 9Ø
FD 25
RT 9Ø
FD 5Ø
BK 1ØØ
HOME
```

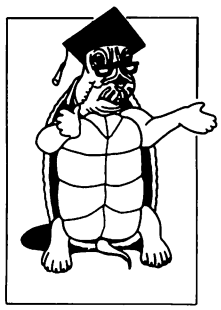
Describe what happens. _____

```
CS
LT 9Ø
FD 25
RT 9Ø
FD 5Ø
BK 1ØØ
PU
HOME
PD
```

Describe what happens. _____

```
RT 9Ø
FD 25
LT 9Ø
FD 5Ø
BK 1ØØ
PU HOME PD
HT
```

Describe what happens. _____



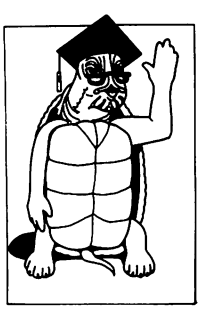
Turtle Talk. PENUP, PENDOWN, and PENERASE are additional commands that control the drawing mode in Turtle Graphics. The turtle's PEN is always DOWN and ready to draw lines when the turtle enters the screen.

Parts 1, 2, and 3 demonstrate that when PENUP (abbreviated PU) is entered, the turtle's pen is no longer in position to draw. It allows relocation of the turtle on the screen without drawing. Once the PENUP command is issued, there is no drawing until the order is reversed by entering PENDOWN (abbreviated PD).

In Part 5, PENERASE (abbreviated PE) changes the drawing mode so that any line or dot in the path of the turtle is erased. It remains in erase mode until PD is entered. A common error when using PU or PE is forgetting to return to the PD mode.

CLEAN, CS, and HOME are other useful commands. Although similar, each performs a distinct function. CLEAN, which has no abbreviation, erases the drawing on the screen but does not affect the position and heading of the turtle. CS erases all drawing and also returns the turtle to the HOME position.

When the command HOME is entered, the turtle returns to its HOME position, drawing a path and leaving all previously drawn lines on the screen. To avoid leaving an unwanted path, precede the command HOME with PU and follow it with PD. Although it is not necessary to type these commands on one line, it helps to visualize them as a unit performing one function. The recommended format to complete a drawing is to conclude with this series of commands. It provides uniformity and may prevent errors.



Turtle Truth. Study the following commands and predict what and where the turtle is to draw. Check your prediction by entering the commands.

```
CS
LT 45
FD 7Ø
SETH Ø
PU
BK 8Ø
PD
```

```
CLEAN
FD 8Ø
RT 9Ø
FD 4Ø
RT 9Ø
FD 8Ø
RT 9Ø
FD 4Ø
PE
BK 4Ø
RT 9Ø
PD
FD 4Ø
LT 9Ø
FD 4Ø
PU HOME PD
```

Turtle Tester. Draw the letter B as shown in Figure 1-5. (Answer on page 230.)

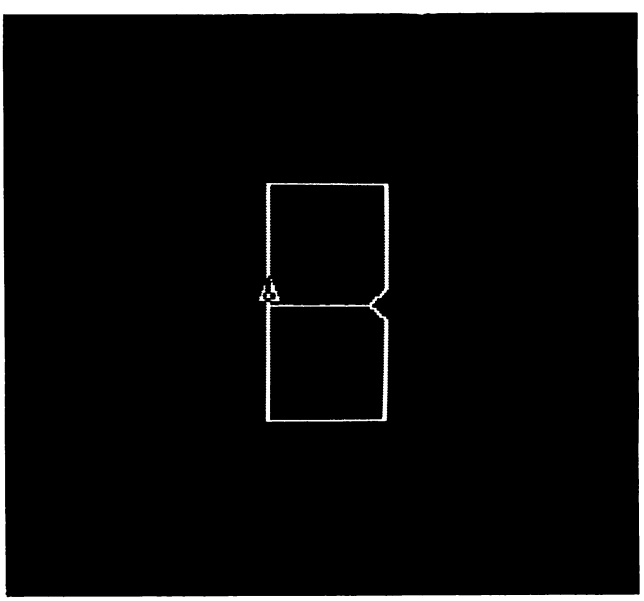
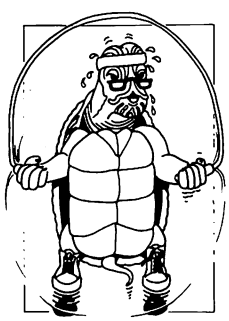


Figure 1-5.

Turtle Teaser. Draw your initials on the screen in an interesting arrangement.

Activity 1.4 Turtle Repetitions



Turtle Trials. This activity introduces a technique that simplifies the writing of repetitive commands and demonstrates how to draw regular polygons.

1. Enter the following commands and comment on each section.

```
CS
ST
FD 5Ø
RT 9Ø
FD 5Ø
RT 9Ø
FD 5Ø
RT 9Ø
FD 5Ø
RT 9Ø
```

What diagram is drawn on the screen? _____
What direction is the HEADING? _____
What two commands are repeated? _____
How many times are those commands repeated? _____

2. On the Apple II Plus, the [(left bracket) is obtained by pressing the SHIFT key and the N key at the same time. The] (right bracket) is obtained by pressing the SHIFT key and the M key. On the Apple IIe and the Franklin Ace 1000, the brackets are available on the standard

keyboard. These brackets must be used with the REPEAT command. Enter the following command.

```
REPEAT 4 [FD 5Ø RT 9Ø]
```

What diagram is on the screen? _____

Count the corners. _____

What is the product of the number of corners and the size of one angle (4*90)? _____

NOTE The symbol for multiplication is *.

3. Enter the following command and observe the screen display.

```
PE
REPEAT 4 [FD 5Ø RT 36Ø/4]
```

Describe what happens. _____

NOTE The symbol for division is /.

4. Enter the following commands and describe the effect of each command.

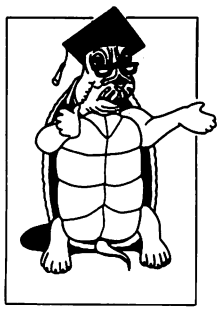
```
PD _____
REPEAT 3 [FD 8Ø LT 36Ø/3] _____
PR 36Ø/3 _____
PE _____
REPEAT 3 [FD 8Ø LT 12Ø] _____
PD _____
```

5. Enter the following commands and describe what happens.

```
REPEAT 6 [FD 5Ø RT 36Ø/6] _____
PR 36Ø/6 _____
FD 5Ø _____
```

6. Enter these commands and record the effect each one has on the screen display.

```
CS
REPEAT 7 [FD 5Ø LT 36Ø/7] _____
PR 36Ø/7 _____
PR 36Ø/8 _____
CS
REPEAT 8 [FD 4Ø LT 45] _____
```



Turtle Talk. A polygon is a closed figure consisting of three or more connected line segments. Polygons are classified by their number of sides. Names of polygons with three to ten sides are as follows: triangle (three sides), quadrilateral (four sides), pentagon (five sides), hexagon (six sides), heptagon (seven sides), octagon (eight sides), nonagon (nine sides), and decagon (ten sides).

A theorem from geometry states that the sum of the exterior angles of any polygon is always 360 degrees. Figure 1-6 illustrates that an exterior angle (that is, angle A) is formed by extending a side of a polygon (dotted line), and considering the angle formed by that side and the adjacent side. Other exterior angles (angles B, C, D, E) are formed in a similar manner by extending each of the sides in succession. There are as many exterior angles for a polygon as the number of sides. In contrast to exterior angles, interior angles are the remaining angles "inside" the polygon.

A polygon is regular if it is equilateral (all sides the same length) and equiangular (all angles the same measure). If the polygon is regular, it can be shown that the exterior angles are equal. Thus, in Figure 1-6, the measure of angle A is 72 degrees (360/5), since the regular pentagon has five sides.

Figure 1-7 displays some other common regular polygons and their corresponding exterior angle measures. Notice the square is the only regular polygon where the interior and exterior angles have the same measure.

Parts 2, 3, and 4 demonstrate that the turtle forms a polygon by completing an entire 360 degree rotation. Since the turtle's revolution created by the RT and LT commands corresponds to the exterior angles of the polygon, dividing 360 by the number of sides of the polygon determines the appropriate values for RT or LT. Equivalently, the product of the number of angles times the size of one angle will equal 360. Seymour Papert refers to this as the "TOTAL TURTLE TRIP THEOREM." This principle is useful when drawing any regular polygon. It allows the user to rely on the computer to calculate the size of exterior angles by the quotient of 360/N, where N is the number of sides of the regular polygon.

By using a combination of this principle and the REPEAT command, the user has an efficient method of drawing regular polygons. The REPEAT command provides a convenient way to combine groups of statements that occur more than once. This command requires a numerical value before the square brackets ([]) to indicate the number of times the sequence within the brackets is to be performed. For example, regular polygons are drawn with the following single line commands.

```
REPEAT 4 [FD 5Ø RT 36Ø/4] (SQUARE)
REPEAT 3 [FD 8Ø LT 12Ø] (TRIANGLE)
```

```
REPEAT 6 [FD 5Ø RT 6Ø] (HEXAGON)
REPEAT 7 [FD 5Ø LT 36Ø/7] (HEPTAGON)
```

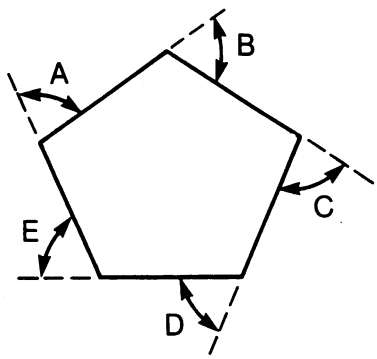


Figure 1-6.

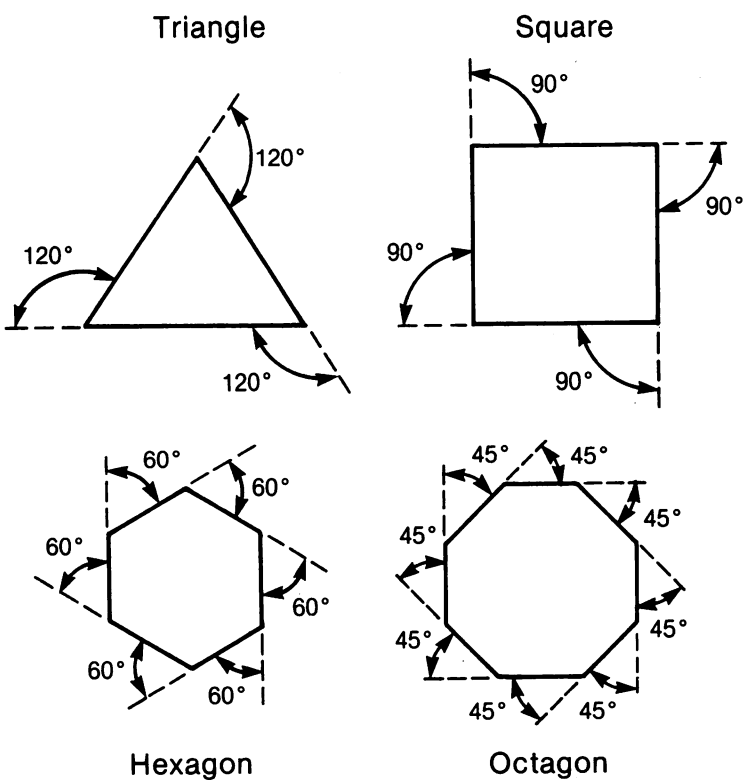
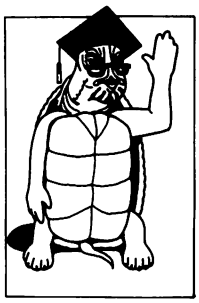


Figure 1-7.



Turtle Truth. Study the following commands and predict what the turtle is to draw. Check your prediction by entering the commands as given.

```
CS
PU
FD 3Ø
PD
REPEAT 8 [FD 4Ø LT 36Ø/8]
REPEAT 8 [FD 4Ø RT 36Ø/8]
PR 36Ø/8
RT 135
FD 4Ø
RT 9Ø
FD 4Ø
SETH Ø
BK 4Ø
REPEAT 8 [FD 4Ø LT 45]
REPEAT 8 [FD 4Ø RT 45]
PU HOME PD
CTRL-L
CTRL-T
CTRL-S
CS
```

Turtle Tester. Make the same arrangement of shapes using hexagons. (Answer on page 230.)

Turtle Teaser. Make the STOP sign illustrated in Figure 1-8.

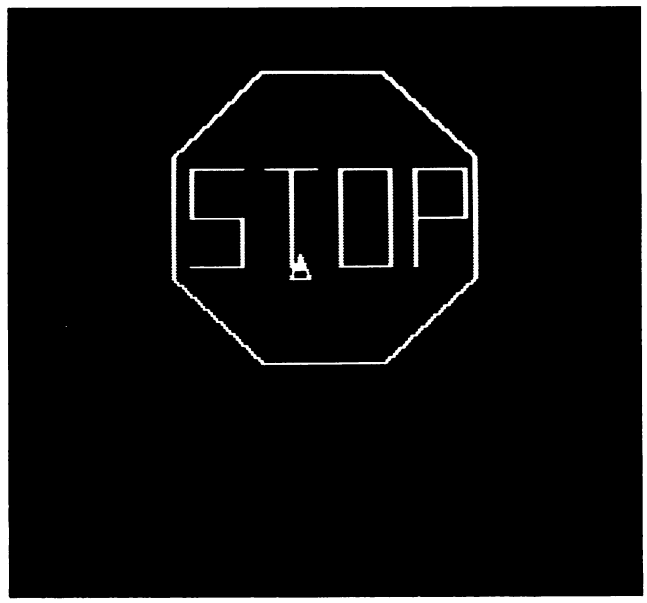
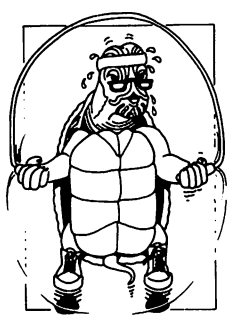


Figure 1-8.

Activity 1.5 Turtle Circles



Turtle Trials. This activity demonstrates how to draw circles.

NOTE Mathematically, a circle is the collection of all points which are the same distance from a common point known as the center. The radius of a circle is a line segment which is drawn from the center to some point on the circle. A diameter, which is twice the length of a radius, is a line segment that passes through the center of the circle and touches two of its points. There are an unlimited number of radii and diameters for each circle. Figure 1-9 illustrates these parts of a circle.

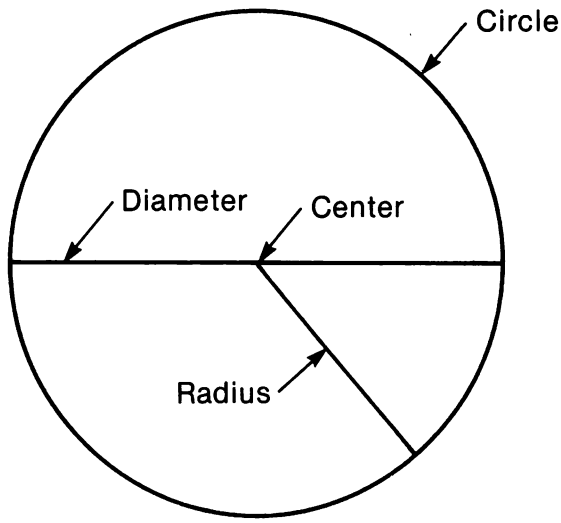


Figure 1-9.

1. Enter the following commands and record your observations on the lines provided.

```
CS
REPEAT 360 [FD 1 RT 1] _____
PE _____
REPEAT 360 [FD 1 RT 360/360] _____
PD
REPEAT 360 [FD 1 LT 1] _____
```

2. Enter the following commands and describe the screen display.

```
CS
CIRCLE _____
CIRCLE R _____
CIRCLER _____
CIRCLER 20 _____
RT 90 _____
FD 20 _____
HT
```

NOTE The input value for CIRCLER is the radius of the circle drawn.

```
FD 20 _____
ST
LT 90 _____
```

```
PE
CIRCLEL 2Ø _____
```

NOTE The line left on the screen is the diameter of the CIRCLEL 20.

```
LT 9Ø _____
FD 4Ø _____
SETH Ø _____
PD
```

3. Enter the following commands and observe the result.

```
CS
CIRCLEL 1Ø
CIRCLEL 2Ø
CIRCLEL 4Ø
CIRCLEL 3Ø
```

Describe the screen display. _____

4. Enter the following commands and record the results.

```
CS
LT 9Ø _____
PU _____
FD 1Ø _____
PD _____
RT 9Ø _____
CIRCLER 1Ø _____
LT 9Ø _____
PU _____
FD 15 _____
PD _____
RT 9Ø _____
CIRCLER 25 _____
LT 9Ø _____
PU _____
FD 2Ø _____
PD _____
RT 9Ø _____
CIRCLER 45 _____
LT 9Ø _____
PU _____
FD 25 _____
PD _____
```

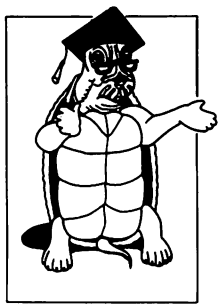
```
RT 9Ø _____
CIRCLER 7Ø _____
LT 9Ø _____
PU _____
FD 3Ø _____
PD _____
RT 9Ø _____
CIRCLER 1ØØ _____
PU HOME PD _____
CTRL-L _____
CTRL-S _____
```

5. Type the following and observe the screen display.

```
CS _____
CIRCLEL 4Ø _____
CIRCLER 4Ø _____
PU _____
FD 8Ø _____
PD _____
CIRCLER 4Ø _____
CIRCLEL 4Ø _____
PU HOME PD _____
HT _____
ST _____
```

6. Enter the commands and record the results.

```
CS
REPEAT 1Ø [CIRCLEL 3Ø RT 36Ø/1Ø] _____
CS
REPEAT 1Ø [CIRCLER 3Ø LT 36] _____
CTRL-L _____
CTRL-S _____
CS
REPEAT 4 [CIRCLEL 8Ø LT 36Ø/4] _____
CS
REPEAT 7 [CIRCLER 4Ø RT 2Ø] _____
CS
REPEAT 5Ø [CIRCLEL 4Ø RT 36Ø/5Ø] _____
```



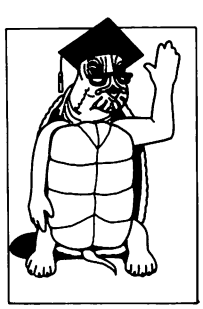
Turtle Talk. In graphics, a circle is represented as a regular polygon with many sides. The precision of the circle increases as the number of the sides of the polygon increases.

Part 1 of this activity demonstrates that with Apple Logo, a circle can be represented as a regular polygon consisting of 360 sides in the procedure REPEAT 360 [FD 1 RT 1]. The circle is formed by repeating the steps, FD 1 RT 1, until the figure is closed. The forward movement of one turtle unit followed by a one degree rotation, 360 times, completes one revolution, thus satisfying the TOTAL TURTLE TRIP THEOREM.

The Logo diskette includes two procedures for drawing circles (36-sided polygons) in the STARTUP file. Remember, the STARTUP file is loaded when the system is booted. CIRCLERight draws the circle clockwise. CIRCLELeft draws the circle counterclockwise. Both procedures require the input of the radius of the desired circle.

By entering CIRCLER 20 in Part 2, a circle with a radius of 20 units is drawn in a clockwise fashion. Once the circle is completed, the sequence [RT 90 FD 20] results in the turtle drawing a radius and stopping at the center of the circle. Another FD 20 command draws a diameter of the circle which passes through the unseen central point to an opposite point on the perimeter. This divides the circle in half.

Parts 4, 5, and 6 demonstrate a variety of circular designs that can be achieved by using the CIRCLER and CIRCLEL commands. Of particular interest is the use of the TOTAL TURTLE TRIP THEOREM to obtain interesting circular effects by consecutively rotating the turtle prior to drawing a circle. In order to complete the circular patterns, the number of degrees in each rotation is 360/N, where N is the number of times the drawing of the circle is repeated.



Turtle Truth. Study the following commands and predict the results. Enter the commands to confirm your prediction.

CS _____

RT 9Ø _____

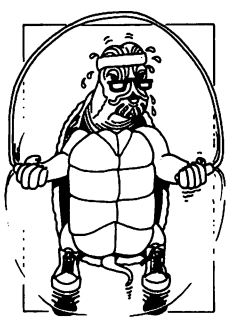
FD 7Ø _____

```
CIRCLER 30
RT 90
FD 30
REPEAT 12 [FD 30 BK 30 RT 360/12]
LT 90
BK 30
LT 90
PU
FD 30
PD
FD 15
RT 90
FD 7
BK 14
FD 7
RT 90
FD 15
RT 90
FD 90
SETH 90
CIRCLER 30
RT 90
FD 30
REPEAT 12 [FD 30 BK 30 RT 360/12]
LT 180
FD 30
RT 30
FD 50
SETH 135
FD 20
SETH 90
FD 10
BK 10
LT 135
FD 20
RT 90
FD 20
SETH 90
FD 10
PU HOME PD
HT
```

Turtle Tester. Use the comprehensive example drawing and add more detail to the design of the vehicle.

Turtle Teaser. Design the exterior of an expensive van or RV.

Activity 1.6 Turtle Arcs



Turtle Trials. This activity demonstrates how to draw arcs.

NOTE An arc is an incomplete or partially constructed circle. The length of an arc is measured in degrees as a fractional part of the circumference of the circle with a given radius. For example, an arc measure of 180 degrees is 180/360 or 1/2 of the circumference of a circle.

1. Enter the commands and record your observations on the lines below.

CS _____

CIRCLER 5Ø _____

RT 9Ø _____

FD 5Ø _____

```
FD 5Ø _____
LT 9Ø _____
PE _____
ARC _____
ARCL _____
ARCL 5Ø _____
ARCL 5Ø 18Ø _____
ARCL 5Ø 9Ø _____
ARCL 5Ø 1Ø _____
ARCL 5Ø 8Ø _____
PD _____
```

NOTE The input 50 refers to the radius of the circle from which the arc is taken. The second input value refers to the measure of the arc that is desired.

2. Enter these commands and describe the results on the lines below.

```
CS
ARCR 75 12Ø _____
RT 6Ø _____
ARCR 75 12Ø _____
RT 6Ø _____
```

What is the sum of the number of degrees in each arc and the number of degrees in each RT turn? _____

```
PR 12Ø + 12Ø + 6Ø + 6Ø _____
REPEAT 2 [ARCR 75 12Ø RT (18Ø - 12Ø)] _____
PR 12Ø + (18Ø - 12Ø) _____
```

NOTE Parentheses are used to group arithmetic operations. Square brackets are used to indicate the start and end of Logo commands or lists.

3. Enter the following commands and record your observations.

```
CS
REPEAT 1Ø [ARCL 14 18Ø LT 18Ø] _____
RT 18Ø _____
```

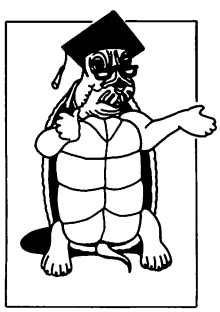
4. Enter the following commands and observe the screen display.

```
CS
REPEAT 4 [ARCR 3Ø 18Ø LT 9Ø] _____
ARCR 3Ø 9Ø _____
```

```
CIRCLER 6Ø _____
CTRL-L _____
CTRL-S _____
```

5. Type the following commands and record the results.

```
CS
REPEAT 9 [ARCL 15 9Ø ARCR 15 9Ø ARCL 15 9Ø
  ARCR 15 9Ø RT 16Ø] _____
HT
```



Turtle Talk. The Logo diskette contains two procedures for drawing circular arcs in the STARTUP File. The ARCRight (ARCR) and ARCLeft (ARCL) commands require two input values. The first value defines the radius of the arc and the second value determines the length of the arc in arc degrees. Part 1 demonstrates that successive ARCL commands with a common radius (50), but with arc degree measures which total 360 degrees (180 + 90 + 10 + 80), eventually draw a circle. Note that the sequence of commands is equivalent to the CIRCLEL 50 command.

The ARCR and ARCL commands are quite useful for drawing petal-like diagrams. The important principle to remember when drawing these shapes is the TOTAL TURTLE TRIP THEOREM. The total sum of arc degrees necessary to complete the entire petal is 360 degrees. The total sum of arc degrees necessary to complete one side of the petal is 180 degrees.

Figure 1-10 illustrates the relationship of critical turtle headings as the arc in Part 2 is drawn. At position A, the turtle has a heading of zero. Position B indicates that the heading is 120 degrees following the ARCR 75

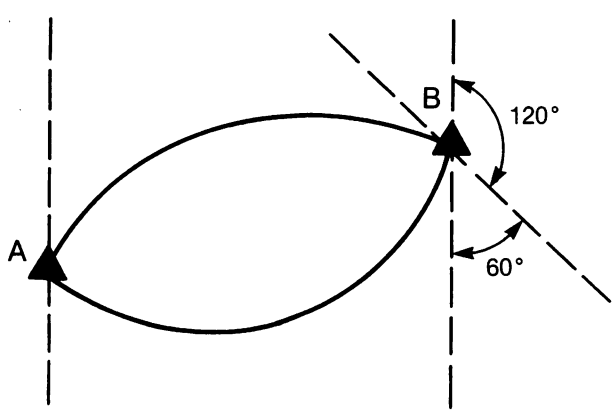
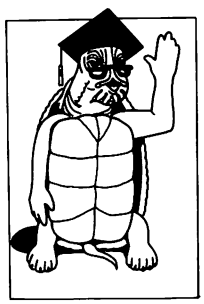


Figure 1-10.

120 command. Before drawing the bottom section of the petal (arc), the turtle heading must become 180, which is accomplished by the RT 60 command. Repeating this process twice (REPEAT 2 [ARCR 75 120 RT 60]) completes the petal and places the turtle in its original heading of zero.



Turtle Truth. Study the following commands and predict what the turtle is to draw. Check your prediction by entering the commands as given.

```
CS
PU
FD 5Ø
RT 9Ø
PD
CIRCLER 5Ø
ARCR 5Ø 45
SETH Ø
REPEAT 2 [ARCR 7Ø 1ØØ RT 8Ø]
PU HOME
FD 5Ø
LT 9Ø
PD
ARCL 5Ø 45
SETH 36Ø
REPEAT 2 [ARCL 7Ø 1ØØ LT 8Ø]
PU HOME PD
```

Turtle Tester. Observe the outline of the animal's face on the screen. Add eyes, eyebrows, a mouth, and whiskers, then return the turtle to HOME to be the nose. Use ARCS and CIRCLES for as many facial features as possible.

Turtle Teaser. Draw an entire animal using several ARCS and CIRCLES.

Reference Point 1

Command Review. The following list contains all of the abbreviations and definitions of the commands introduced thus far.

Abbreviations	Definitions
ST	Show Turtle.
HT	Hide Turtle.
FD N	Forward requires an input number N.
BK N	Back requires an input number N.
CS	Clear Screen and return turtle to HOME.
CTRL-L	Displays full graphics screen.
CTRL-T	Displays full text screen.
CTRL-S	Displays split screen with four lines of text and the remainder of the screen is dedicated to graphics (normal mode).
RT N	Pivots turtle to its right and requires an input number N.
LT N	Pivots turtle to its left and requires an input number N.
SETH N	Points the turtle to the direction specified by the input number N. Input is based on the clockwise revolution of the turtle in relation to HOME.
PR HEADING	Indicates the numerical value of the current SETH direction.
HOME	Positions turtle to the center of the screen at SETH Ø.
PU	Permits movement without drawing.
PD	Places turtle in drawing mode.

Abbreviations	Definitions
PE	Places turtle in erase mode.
CLEAN	Erases drawing but does not affect position or heading of turtle.
REPEAT N [—]	Runs a set of commands as many times as is indicated by the input number N.
	Brackets enclose a set of commands.
PR N	Prints the numerical value of N.
PR [—-]	Prints what is enclosed by the square brackets [ ].
CIRCLER R	A built-in procedure to draw a circle to the turtle's right and requires an input of radius R.
CIRCLEL R	A built-in procedure to draw a circle to the turtle's left and requires an input of radius R.
ARCR R S	A built-in procedure that draws a segment of CIRCLER and requires an input of radius R and an input of arc size S.
ARCL R S	A built-in procedure that draws a segment of CIRCLEL and requires an input of radius R and an input of arc size S.

Application. Enter the following commands and study the resulting picture. Notice the programming techniques employed, especially the use of nested brackets (brackets within brackets) in the second line. With this structure, the [FD 30 RT 90] sequence is performed eight (4*2) times since the REPEAT 4 command happens twice. In contrast, the second FD 30 command is performed only twice; it is part of the REPEAT 2 sequence, but not part of the REPEAT 4 sequence.

```
CS
REPEAT 2 [REPEAT 4 [FD 3Ø RT 9Ø] FD 3Ø]
REPEAT 4 [FD 3Ø RT 9Ø]
BK 3Ø
LT 9Ø
REPEAT 4 [FD 3Ø RT 9Ø]
BK 6Ø
REPEAT 4 [FD 3Ø RT 9Ø]
RT 9Ø
REPEAT 4 [FD 3Ø LT 45 FD 42 LT 45]
LT 135
FD 11
LT 9Ø
REPEAT 4 [ARCR 1Ø 18Ø LT 9Ø FD 11 RT 45 FD 3Ø RT 45 FD 11 LT 9Ø]
```

NOTE The exclamation mark that appears on the screen as you are typing the command that drops to the next line is a "line continuation

prompt." It has no effect on the drawing, but signifies that a string of commands exceeds the amount of characters which can fit across the screen.

```
RT 90
FD 31
RT 45
FD 11
REPEAT 3 [FD 10 LT 120]
BK 11
RT 90
FD 90
LT 90
REPEAT 3 [FD 30 RT 120]
FD 15
SETH 0
PU
FD 10
PD
CIRCLER 2.5
CIRCLEL 2.5
HT
```

Suggested Projects

Try your hand at producing some of the objects below. Keep a record of the commands you use. In Chapter 2 you will learn how to make these commands into a procedure and save it on a file disk. You will then be able to enter the procedure name and the computer will draw the entire picture for you.

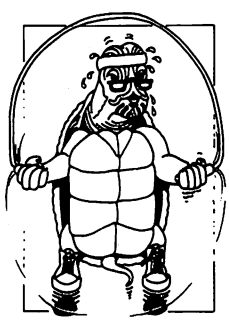
AN IGLOO	SPECTACLES
GREEK COLUMNS	A CACTUS
A VEHICLE	NUMERALS
ICE CREAM SUNDAE	CENTIPEDE
TELEVISION	SAILBOAT
COMPASS	DIGITAL CLOCK

ANYTHING ELSE THAT STRIKES YOUR FANCY

2

Preserving Turtle Tracks

Activity 2.1 Turtle Procedure



Turtle Trials. This activity teaches the user how to initialize a file disk and write procedures in definition mode.

1. You will now initialize a blank disk. Cold boot the system with the APPLE DOS 3.3 SYSTEM MASTER as described in Activity 1.1. After the disk drive light goes off, remove the SYSTEM MASTER and replace it with the blank disk you wish to initialize. Type the following:

   ```
   INIT HELLO
   ```

   You will hear several strange noises. Do not be alarmed. The disk is being initialized or formatted. When the cursor returns, type

   ```
   CATALOG
   ```

   The file HELLO should be listed. You now have a properly initialized disk to use in Part 2 of this activity.

2. Remove the newly initialized diskette, insert the Logo diskette, and warm boot the system by typing PR#6 at the prompt (on an Apple IIe, you warm boot by pressing the CONTROL key, the OPEN APPLE key,

and the RESET key simultaneously). When you receive the partial message

```
PRESS THE RETURN KEY TO BEGIN

IF YOU HAVE YOUR OWN FILE DISKETTE,
INSERT IT NOW, THEN PRESS RETURN
```

leave the Logo diskette in the disk drive and press the RETURN key. Logo will then load the STARTUP file into the computer memory. When you receive the message

```
WELCOME TO LOGO
?■
```

remove the Logo disk. Insert the initialized disk you wish to use as a file disk. Then type

```
SAVE "STARTUP "AIDS
```

When the cursor returns, you will have completed the creation of your own file disk.

NOTE Now that you have this file disk to use, you will not have to repeat the previous steps until you need another file disk. In all future activities, please boot the system with the Logo disk, insert this file diskette when the screen display instructs you to do so, then press the RETURN key. It will be assumed that you have completed this booting process prior to each activity.

3. Until now, you have used built-in commands such as FD, RT, and REPEAT. These commands are commonly called "primitives." You will now learn how to develop "procedures" by using the TO command. The TO command changes the prompt and puts Logo in "definition mode," which allows you to teach the computer a procedure and give the procedure a name. Procedures can then be used like primitives. Enter the following commands and observe the screen display. Record your observations on the lines provided.

```
REPEAT 4 [FD 7Ø RT 9Ø] _____
CS
SQUARE _____
TO SQUARE _____
REPEAT 4 [FD 7Ø RT 9Ø] _____
END _____
SQUARE _____
CS
REPEAT 4 [SQUARE RT 36Ø/4] _____
TO FOUR.SQUARE _____
```

```
REPEAT 4 [SQUARE RTØ] _____
← _____
CTRL-G _____
```

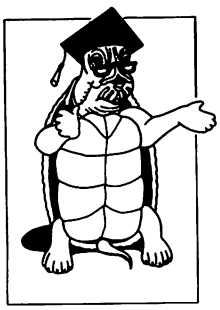
NOTE The command RT90 was an intentional mistake and once RETURN was pressed it could not be altered. By entering CTRL-G, the procedure we were about to define was interrupted or aborted, allowing for a new start. CTRL-G can be used whenever you "Goof" and desire to start over. More sophisticated methods of correcting mistakes will be introduced later in this chapter.

```
TO FOUR.SQUARE _____
REPEAT 4 [SQUARE RT 9Ø]
END
FOUR.SQUARE _____
CS

REPEAT 8 [SQUARE RT 36Ø/8] _____
CS
TO EIGHT.SQUARE _____
REPEAT 8 [SQUARE RT 45]
END
EIGHT.SQUARE _____
```

4. Type the following commands. Notice what happens and record the results.

```
CS
TO TRIANGLE
REPEAT 3 [FD 8Ø LT 12Ø]
END
TRIANGLE _____

CS
REPEAT 7 [TRIANGLE LT 36Ø/7] _____
TO SEVEN.TRI
REPEAT 7 [TRIANGLE LT 36Ø/7]
END
CS
SEVEN.TRI _____

CS
REPEAT 16 [TRIANGLE RT 36Ø/16] _____
```

```
TO SIXTEEN.TRI
REPEAT 16 [TRIANGLE RT 360/16]
END
CS
SIXTEEN.TRI
```



Turtle Talk. Up to now, you have used the Logo diskette and have ignored the message

```
IF YOU HAVE YOUR OWN FILE DISKETTE,
INSERT IT NOW, THEN PRESS RETURN
```

You have now prepared a file diskette on which material developed in subsequent activities will be saved for future use. This second diskette has been "initialized" for Logo. Parts 1 and 2 guide you through the initialization process. These steps must be followed anytime you wish to make a new diskette on which to store information.

Part 2 transfers a file called STARTUP from the Logo diskette to your new diskette. This transfer makes the procedures CIRCLER, CIRCLEL, ARCR, and ARCL available for easy access. You encountered these commands in Activities 1.5 and 1.6.

In Chapter 1, all commands that are entered draw the diagram once and must be retyped each time you want the diagram to be drawn. REPEAT 4 [FD 70 RT 90] in Part 3 serves as a review example, where this command

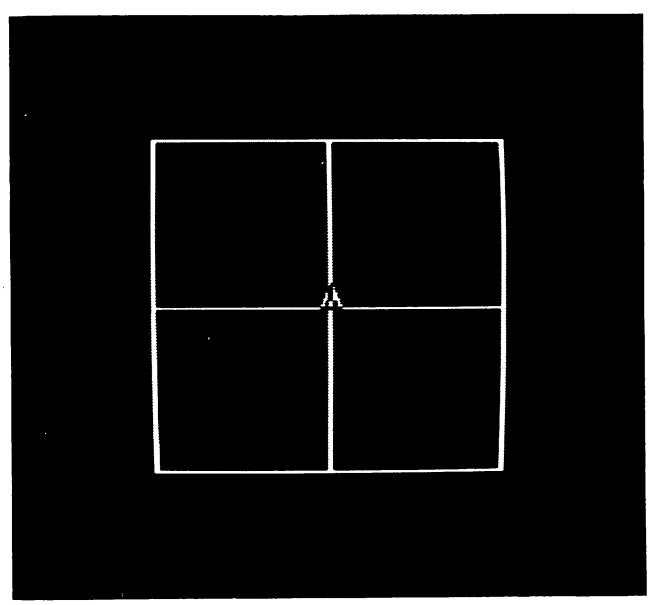


Figure 2-1.

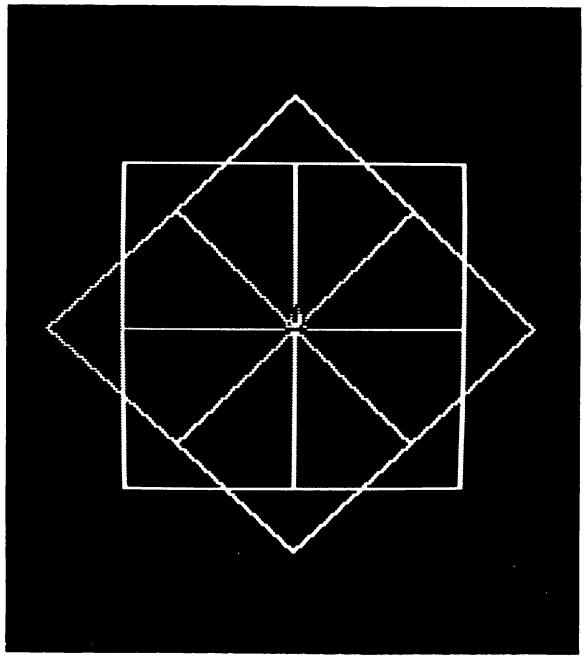


Figure 2-2.

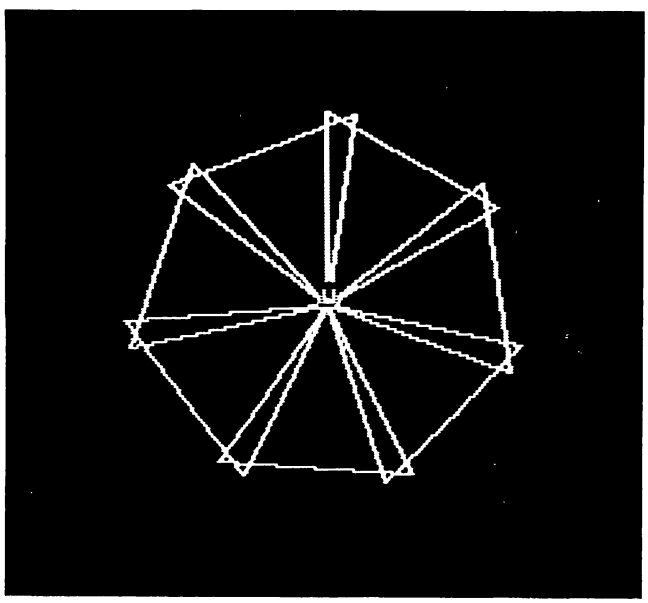


Figure 2-3.

draws a square only once. The command must be re-entered each time a square is to be drawn.

In contrast, the TO command, introduced in Part 3, puts the user in

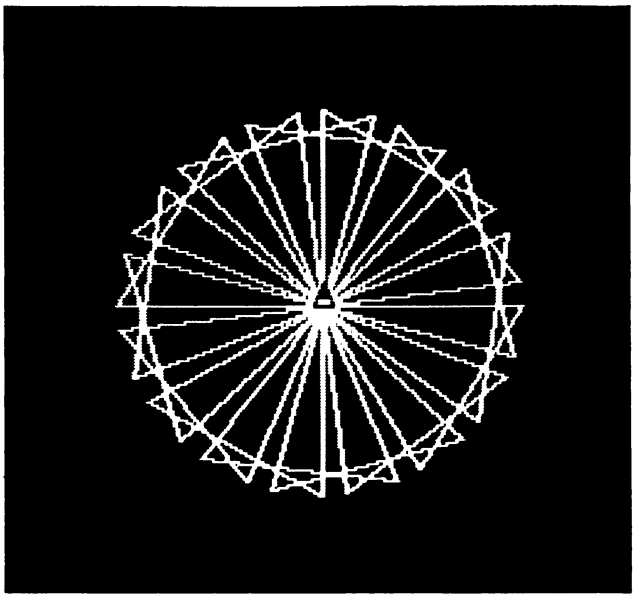
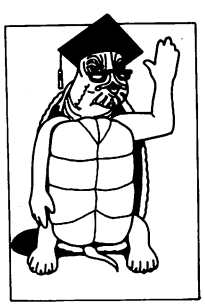


Figure 2-4.

"definition mode" where procedures can be developed. Defining a procedure means that you give a collection of commands a name. Now, by simply typing this name, the commands are performed.

Part 4 defines the various procedures that are introduced in the activity. Four of these procedures, FOUR.SQUARE (Figure 2-1), EIGHT.SQUARE (Figure 2-2), SEVEN.TRI (Figure 2-3) and SIXTEEN.TRI (Figure 2-4) illustrate the principle of the TOTAL TURTLE TRIP THEOREM by drawing rotations of regular polygons defined by SQUARE and TRIANGLE. For example, SIXTEEN.TRI rotates TRIANGLE 360/16 degrees 16 times, which completes the total 360 degree drawing.



Turtle Truth. Study the following commands and predict the results. Type the commands, verify your predictions, and compare the screen display to the picture illustrated in Figure 2-5.

```
CS
FD 3Ø
RT 9Ø
FD 75
RT 9Ø
```

```
FD 3Ø
RT 9Ø
FD 75
RT 9Ø

TO RECTANGLE
CTRL-T
REPEAT 2 [FD 3Ø RT 9Ø FD 75 RT 9Ø]
END
CTRL-S
CS
RECTANGLE
REPEAT 1Ø [RECTANGLE RT 36Ø/1Ø]
TO TEN.REC
REPEAT 1Ø [RECTANGLE RT 36]
END
CS
TEN.REC
```

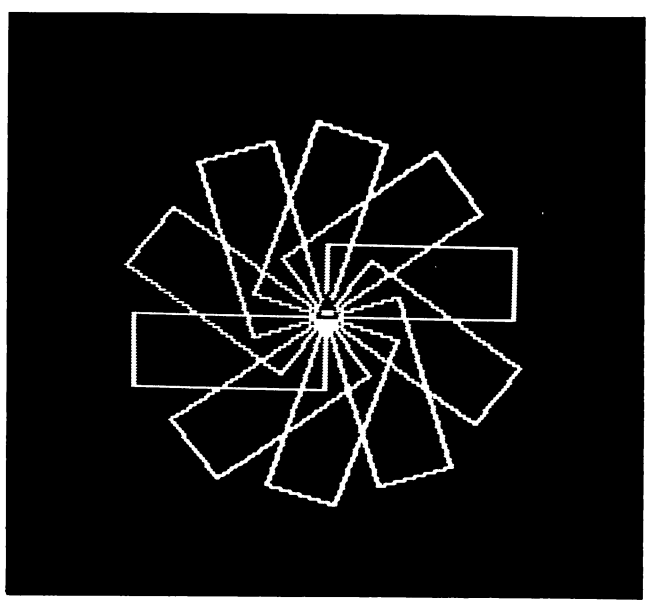


Figure 2-5.

Turtle Tester. The picture illustrated in Figure 2-6 is an example of a PENTAGON that has been rotated six times. Define a procedure called PENTAGON. Now define a procedure that rotates the PENTAGON a number of times to achieve a pleasing symmetrical design. (Answer on page 230.)

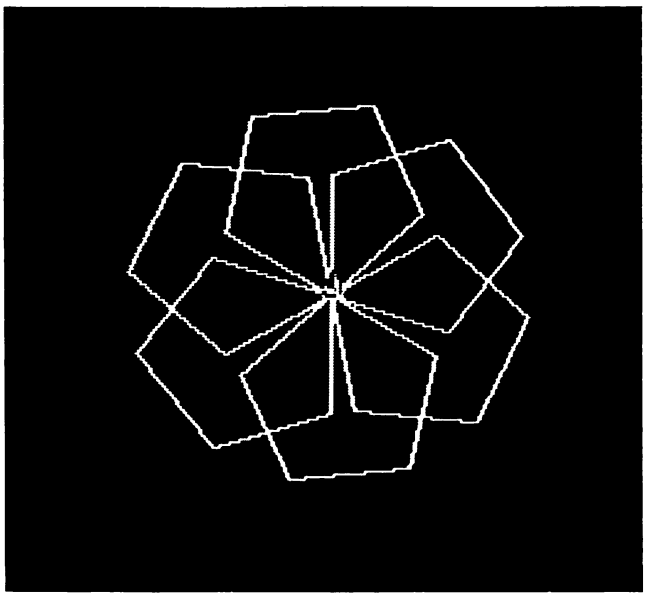


Figure 2-6.

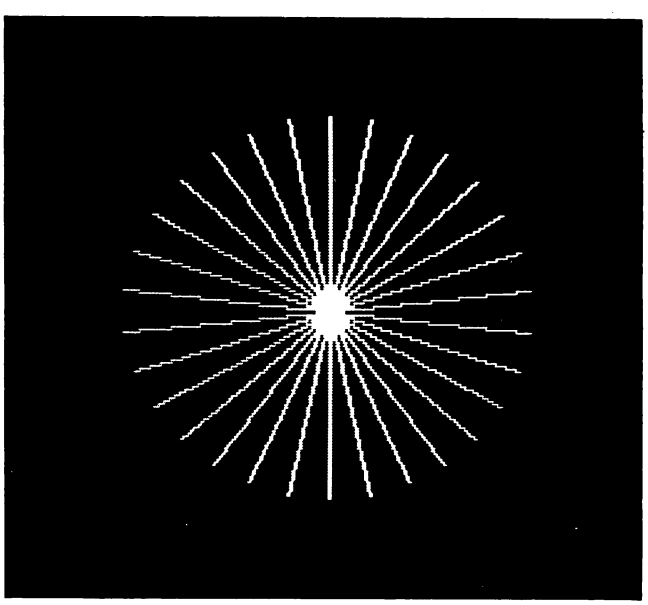
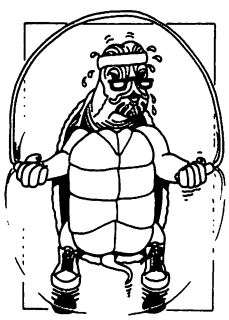


Figure 2-7.

Turtle Teaser. Define a procedure to draw the picture illustrated as Figure 2-7.

Activity 2.2 Turtle Files



Turtle Trials. This activity reinforces the use of the definition mode to write procedures and then teaches you how to save them to a file disk for further use.

1. Cold boot the system with the Logo disk, insert your file diskette when the screen display requests you to do so, and press the RETURN key. When the cursor returns, type

   ```
   CATALOG
   ```

   You should now see a list of files on the disk, including HELLO and STARTUP.LOGO.

2. Enter the following commands. Record the results after each command entered.

   ```
   TO SQUARE _____
   REPEAT 4 [FD 3Ø RT 9Ø] _____
   END _____

   CS
   SQUARE _____
   FD 3Ø _____

   SQUARE _____
   FD 3Ø _____
   ```

```
SQUARE _____
FD 30 _____

CTRL-T _____
CTRL-S _____
```

Review the commands just entered. How many times is the command SQUARE FD 30 entered? _____

_____

```
TO 3SQUARE _____
REPEAT 3 _____
← _____
CTRL-G _____

TO 3SQUARE _____
REPEAT 3 [SQUARE FD 30] _____
END _____
CS _____
3SQUARE _____
```

3. Enter the following commands and record the results.

```
BK 30
RT 90 _____
BK 30 _____
3SQUARE _____
CTRL-T _____
```

NOTE This view of the previously typed text allows the user to observe and copy the commands that were used to achieve the desired screen display when writing a new procedure.

```
TO CROSS.SQUARE _____
3SQUARE _____
BK 30 _____
RT 90 _____
BK 30 _____
3SQUARE _____
END _____
CTRL-S _____
CS
CROSS.SQUARE _____
```

4. Enter the following commands and describe what happens.

```
RT 90 _____
FD 30 _____
RT 45 _____
FD 30 _____
HT _____
FD 10 _____
ST _____
RT 45 _____
FD 30 _____
RT 45 _____
FD 40 _____
RT 45 _____
FD 30 _____
CS
```

5. Correct your previous estimations by typing the following commands. Record your observations.

```
CROSS.SQUARE _____
RT 90 _____
FD 30 _____
RT 45 _____
FD 40 _____
FD 2 _____
HT _____
ST _____
```

NOTE The back of the turtle appears dark at the exact point it crosses the white line.

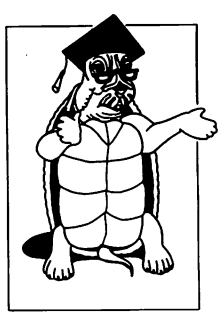
```
RT 45 _____
FD 30 _____
RT 45 _____
FD 42 _____
RT 45 _____
FD 30 _____
CTRL-T _____
TO PERIMETER _____
RT 90
REPEAT 4 [FD 30 RT 45 FD 42 RT 45]
END _____
CTRL-S _____
```

6. Enter the following commands and describe the results.

```
CS
CROSS.SQUARE _____
PERIMETER _____
TO SHELL _____
CROSS.SQUARE _____
PERIMETER _____
END _____
CS _____
SHELL _____
```

7. In order to reuse these procedures after shutting down the computer, we must save them on our file disk. Enter the following and observe the results.

```
SAVETURTLE.PARTS _____
SAVE TURTLE.PARTS _____
SAVE "TURTLE.PARTS _____
SHELL _____
CS _____
ERPS _____
SHELL _____
```



Turtle Talk. In Activity 2.1, the concept of a procedure was introduced. This activity elaborates on the procedural approach to programming through the development of an example.

Logo is considered to be a "procedural" programming language. Procedures are small programs that collectively build a larger program. They allow for the breaking up of a complex figure into a number of small manageable parts. Using Logo terminology, procedures define new words or commands in terms of the primitives provided by the language itself. Procedures provide the possibility for a systematic approach to computer problem-solving. They are designed to minimize effort when finding and correcting logical errors.

In this activity, the complex figure (the shell of the turtle) is created by breaking the components of the shell into sub-units, which are used repetitively.

The most primitive piece of the turtle shell is a square. Upon defining a procedure for drawing a SQUARE, three vertically adjacent squares are displayed and subsequently defined as a 3SQUARE procedure in Part 3.

In turn, Part 4 demonstrates that 3SQUARE is used in the CROSS.SQUARE procedure to achieve the pattern desired for the top of the turtle shell, thereby completing the sequence of building components for the desired diagram.

In Part 6, PERIMETER defines a procedure to connect all exterior vertices of the cross, thus completing the shape of the shell, which is illustrated in Figure 2-8.

The simplified procedure for SHELL is defined in Part 7, and requires only the use of CROSS.SQUARE and PERIMETER to complete the drawing of the turtle shell.

These newly created procedures are saved on disk for later access by the use of the SAVE command. TURTLE.PARTS is the arbitrary name given to the file that includes the five procedures you defined: SQUARE, 3SQUARE, CROSS.SQUARE, PERIMETER, and SHELL. Since Logo only recognizes one-word file names, a period can be used in place of a blank space for multi-word names.

The ERPS command ERases all ProcedureS currently in the computer's memory in what is called the "workspace." After ERPS has been used, any additions to this file or use of these procedures requires the user to retrieve this file from the disk and enter it into the workspace. This is an important topic in the next activity.

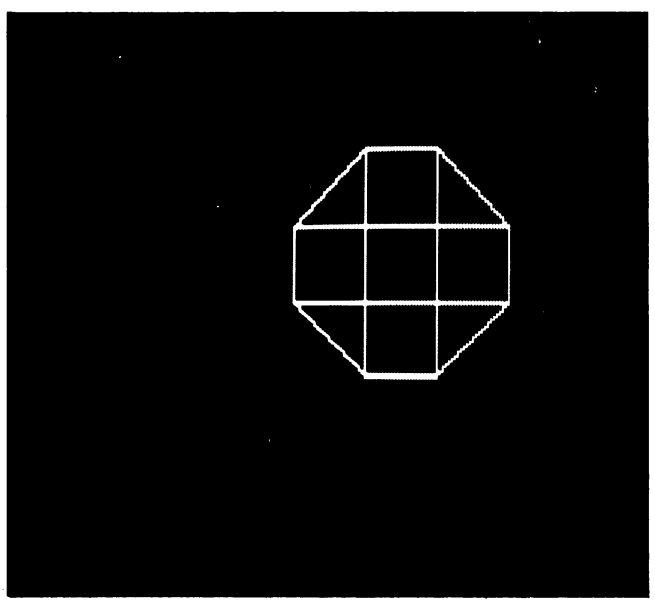
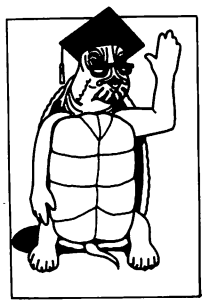


Figure 2-8.



Turtle Truth. Study the following commands and predict the result. Enter the commands to confirm your prediction and compare the screen display with the picture illustrated in Figure 2-9.

```
TO HEAD
PU
FD 5Ø
RT 9Ø
PD
CIRCLER 5Ø
END
HEAD

ARCR 5Ø 45
SETH Ø
REPEAT 2 [ARCR 7Ø 1ØØ RT 8Ø]

CTRL-T
TO RIGHT.EAR
ARCR 5Ø 45
SETH Ø
REPEAT 2 [ARCR 7Ø 1ØØ RT 8Ø]
END

CTRL-S
CS
HEAD
RIGHT.EAR

PU
HOME
FD 5Ø
LT 9Ø
ARCL 5Ø 45
SETH 36Ø
PD
REPEAT 2 [ARCL 7Ø 1ØØ LT 8Ø]
CTRL-T
TO LEFT.EAR
PU
HOME
FD 5Ø
LT 9Ø
ARCL 5Ø 45
SETH 36Ø
```

```
PD
REPEAT 2 [ARCL 7Ø 1ØØ LT 8Ø]
END

CTRL-S
CS
HEAD
RIGHT.EAR
LEFT.EAR
PU HOME PD

TO WABBIT
HEAD
RIGHT.EAR
LEFT.EAR
PU HOME PD
END

CS
WABBIT
SAVE "WABBIT.PIECES
CS
WABBIT
ERPS
WABBIT
```

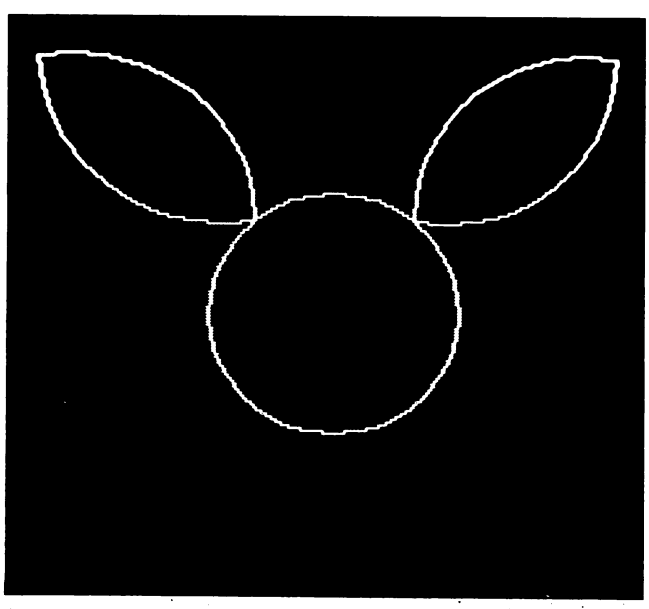
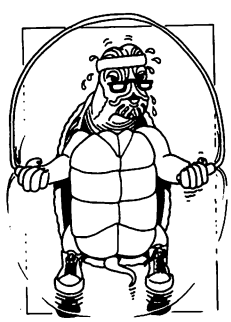


Figure 2-9.

Turtle Tester. Refer back to Activity 1.4 and define several procedures to draw a triangle, pentagon, hexagon, heptagon, octagon, nonagon, and decagon. Save these procedures in a file called REGULAR.POLYGONS. When you are done, erase the procedures from the workspace. Catalog the file disk to verify the existence of this new file. (Answer on page 231.)

Turtle Teaser. Clear the screen. Write procedures to draw uppercase letters. Save them under the file name CAPS. When you are done, erase the procedures from the workspace. The ambitious reader should be warned that this project could quickly exhaust the available memory in the workspace if the entire alphabet were to be created.

Activity 2.3 Turtle Inventory



Turtle Trials. This activity teaches the user how to retrieve a file from a previously created file disk and manipulate the procedures in that file.

1. Cold boot the computer with the Logo diskette and file diskette as usual.

2. When you receive the message

```
WELCOME TO LOGO
?■
```

   type CATALOG and press RETURN.

NOTE If you saved all the procedures in Activity 2.2 under the suggested file names, you will see the following:

```
DISK VOLUME 254

A    # HELLO
T    # STARTUP.LOGO
T    # TURTLE.PARTS.LOGO
T    # WABBIT.PIECES.LOGO
T    # REGULAR.POLYGONS.LOGO
T    # CAPS.LOGO
```

The # sign preceding each file stands for a number that indicates the amount of space occupied by each file on the diskette. The magnitude of the number may vary depending on how many procedures are in the file and the length of each procedure. The "T" represents a Text File. The "A" represents Applesoft, a BASIC language program.

3. Type the following commands. Wait for the cursor to reappear after each entry. Observe and record the results.

```
LOAD TURTLE.PARTS.LOGO _____
LOAD "TURTLE.PARTS.LOGO _____
LOAD "TURTLE.PARTS _____
TURTLE.PARTS _____
POTS _____
```

NOTE POTS stands for Print Out TitleS.

```
SQUARE _____
CS
3SQUARE _____
CS
CROSS SQUARE _____
POTS _____
CTRL-T _____
CROSS.SQUARE _____
CS
PERIMETER _____
CS
CROSS.SQUARE _____
PERIMETER _____
CS
SHELL _____
```

4. Enter the following commands and record the results.

```
ERPS _____
POTS _____
```

```
CATALOG _____
LOAD "WABBIT.PIECES _____
POTS _____
WABBIT _____
LOAD "TURTLE.PARTS _____
POTS _____
SHELL _____
PU HOME PD _____
WABBIT _____
ERPS _____
CS _____
```

NOTE ERPS stands for ERase ProcedureS.

5. Enter the following commands and describe the screen display.

```
POTS _____
LOAD "TURTLE.PARTS _____
POTS _____
POPS _____
```

NOTE POPS stands for Print Out ProcedureS.

```
CLEARTEXT _____
POTS _____
PO "3SQUARE _____
3SQUARE _____
PO "SHELL _____
CTRL-T _____
```

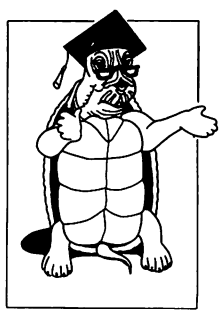
NOTE PO stands for Print Out.

```
ERPS _____
POTS _____
```

6. Enter the following commands and record the results.

```
CATALOG _____
LOAD "TURTLE.PARTS _____
POTS _____
SAVE "TORTOISE.PARTS _____
ERPS _____
POTS _____
CATALOG _____
```

```
LOAD "TORTOISE.PARTS _____________________
POTS _____________________________________
SHELL ____________________________________
ERPS _____________________________________
POTS _____________________________________
CATALOG __________________________________
ERASEFILE "TORTOISE.PARTS ________________
CATALOG __________________________________
LOAD "TURTLE.PARTS _______________________
POTS _____________________________________
```



Turtle Talk. Activities 2.1 and 2.2 demonstrate how to create and save procedural files on diskette. In this activity, your concern is how to retrieve this information in an organized way.

When you boot the computer with the Logo diskette and eventually receive the "WELCOME TO LOGO" message, you have a workspace (or part of the computer memory) to use. Initially this workspace is empty. As you define procedures and variables, this information will accrue in the workspace. This storage is temporary in the sense that when you turn the computer off, all information in the workspace is lost, unless you first save the information on a diskette.

The SAVE command introduced in Activity 2.1 transfers all procedures accumulated in the workspace to diskette. Files may be retrieved from a diskette and put into the workspace by using the LOAD command. In Part 3, LOAD "TURTLE.PARTS takes all procedures contained in the file named TURTLE.PARTS from the diskette and makes them available for use in the workspace.

A problem that frequently confronts the Logo user is a lapse of human memory. It is difficult to remember the names of all files, names of the procedures within the particular files, and finally, the list of commands that define the various procedures. Fortunately, there are several commands to assist the user with managing this information.

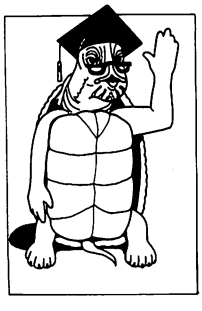
When the user cannot remember the names of files on a diskette, he can use the CATALOG command introduced in Part 4. The CATALOG command provides a list of all Logo file names, which are contained on the diskette. Notice each file name ends with .LOGO. The suffix serves as an indicator that the file is unique to Logo. It is not to be typed when LOADing or SAVEing a file. Files which are of no further use can be removed from the diskette by using the ERASEFILE command. ERASEFILE "TORTOISE.PARTS in Part 6 demonstrates the use of this command.

Once a file has been LOADed into the workspace, POTS (Print Out TitleS) provides a listing of all names or titles of procedures within the workspace at that time. The POTS command is particularly useful when you forget what procedures are contained within a file.

There is one peculiarity of interest. Notice, in Part 4, that the procedures from different files can accumulate in the workspace if successive LOAD commands are used. The LOADing of one file does not erase and replace the contents of the workspace. Instead, the new procedures are appended to the existing procedures. This occurrence may or may not be desirable. In the event you wish an empty workspace prior to LOADing a file, the ERPS (ERase ProcedureS) command is available. Part 5 demonstrates the use of ERPS prior to each LOAD command to assure a clean workspace.

Finally POPS (Print Out ProcedureS) is a command that lists the names and definitions of all procedures contained in the workspace. Part 5 demonstrates one limitation of the POPS command. If the file is lengthy, the listing scrolls off the screen too quickly for viewing. However, PO (Print Out) lets the user observe the listing of commands for a particular procedure, such as PO "SHELL. Also, although not demonstrated in the activity, you can use CTRL-W to freeze the scrolling of text. Pressing any key will cause the listing to continue. The CLEARTEXT command and CTRL-T conveniently clear the screen and devote it entirely to text for a less congested view of the listing.

Complete familiarity with these file and workspace management commands is essential to working efficiently with Logo. Regular use of these functions will help the user develop confidence and expertise.



Turtle Truth. Study the following commands and predict the outcome. Enter the commands to confirm your prediction. Cold boot the system with Logo. When instructed, insert your file diskette.

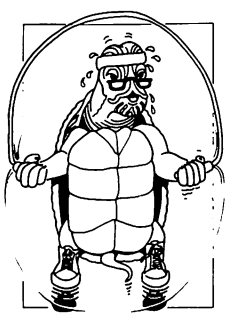
```
CATALOG
LOAD "WABBIT.PIECES
POTS
POPS
WABBIT
PO "WABBIT
CTRL-T
CTRL-L
CTRL-S
SAVE "RABBIT.PARTS
ERPS
```

```
LOAD "RABBIT.PARTS
POTS
WABBIT
ERASEFILE "RABBIT.PARTS
CATALOG
LOAD "WABBIT.PIECES
POTS
CTRL-T
CLEARTEXT
ERPS
POTS
CTRL-S
```

Turtle Tester. Start with the computer off. Cold boot the system with Logo, then insert your file diskette. Call up the catalog of files. Load in the file REGULAR.POLYGONS (created in Activity 2.2). Examine a list of the procedures in that file. Run one of the procedures. Remove the current information in the workspace. Call up the catalog again to verify the existence of the original file.

Turtle Teaser. Perform similar operations on the file CAPS that was created in Activity 2.2.

Activity 2.4 Turtle Editing



Turtle Trials. This activity demonstrates how cursor manipulation can be used to edit existing procedures.

1. Enter the following procedure.

```
TO TRACK
HT
PU
FD 5Ø
RT 9Ø
BK 5Ø
PD
REPEAT 2 [FD 1ØØ ARCR 5Ø 18Ø]
END

TRACK

EDIT "TRACK
```

Notice that the screen display changes in the edit mode. The Logo editor allows the manipulation of the cursor to effect changes in the procedure held in a reserved area of memory known as the "edit buffer."

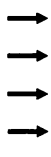
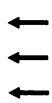
NOTE Throughout the remainder of this activity, you will be learning to make use of CTRL keys for the purpose of editing previously entered Logo commands. The manner in which you enter CTRL commands is different from the manner in which you enter Logo commands. The RETURN key is pressed after entering each Logo command. The RETURN key is not pressed after entering a CTRL command.

Observe and record the results of the following CTRL commands. DO NOT PRESS RETURN AFTER EACH COMMAND OR YOU WILL HAVE UNWANTED MOVEMENT OF THE TEXT.

CTRL-N _____
CTRL-N _____
CTRL-N _____
CTRL-P _____
CTRL-P _____
CTRL-O _____
CTRL-N _____
CTRL-N _____
CTRL-O _____
CTRL-P _____
CTRL-P _____
CTRL-P _____
CTRL-G _____
EDIT "TRACK _____
CTRL-C _____
TRACK

What is the function of CTRL-N? _____
What is the function of CTRL-P? _____
What is the function of CTRL-O? _____
What is the function of CTRL-G? _____
What is the function of CTRL-C? _____

2. Note the movement caused by the following edit commands.

```
EDIT "TRACK
```

```
CTRL-A _____
CTRL-E _____
CTRL-B _____
CTRL-B _____
CTRL-B _____
CTRL-B _____
```

```
CTRL-G _____
TRACK
```

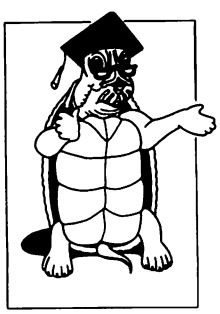
What is the function of →? _____
What is the function of ←? _____
What is the function of CTRL-A? _____
What is the function of CTRL-E? _____
What is the function of CTRL-B? _____

3. Describe the movement produced by each of the following edit commands.

```
EDIT "TRACK
CTRL-N _____
CTRL-D _____
CTRL-D _____
CTRL-N _____
CTRL-N _____
CTRL-K _____
CTRL-Y _____
CTRL-A _____
CTRL-P _____
```

CTRL-K _____
CTRL-N _____
CTRL-N _____
CTRL-O _____
CTRL-Y _____
CTRL-A _____
CTRL-N _____
CTRL-D _____
CTRL-D _____
CTRL-G _____

What is the function of CTRL-D? _____
What is the function of CTRL-K? _____
What is the function of CTRL-Y in conjunction with CTRL-K? _____
_____



Turtle Talk. To alleviate the frustration of modifying procedures, Apple Logo provides several editing commands which allow the user to move the cursor on the screen to make additions and deletions. These editing commands are categorized and summarized below.

Horizontal Control

← moves the cursor one space to the left and erases a character each time the key is depressed.

→ moves the cursor one space to the right each time the key is depressed without erasing any characters.

CTRL-B moves the cursor one space to the left or Backwards without erasing any characters.

CTRL-A moves the cursor to the beginning of the line in which it is positioned.

CTRL-E moves the cursor to the End of the line in which it is positioned.

Vertical Control

CTRL-N moves the cursor down to the Next line from its current position.

CTRL-P moves the cursor up to the Previous line from its current position.

CTRL-O lowers subsequent lines of the procedure to Open a new line at the current cursor position, assuming the cursor is positioned on the first character of a line.

Text Manipulators

CTRL-D Deletes the character under the cursor.

CTRL-K deletes or Kills all characters on line to the right of the cursor.

CTRL-Y inserts or Yanks back the characters which are killed by the most previous CTRL-K.

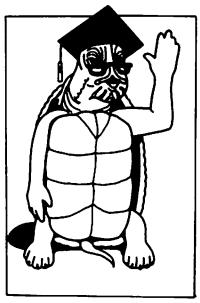
Workspace Management

ER "---------- is used to ERase the designed procedure from the workspace.

Table 2-1 illustrates the various functions and cursor positions for the editing commands discussed in this activity.

Table 2-1. Editing commands with their various functions and cursor positions.

Edit Commands	*Before Using*	*After Using*
←	FORWARD 30■	FORWARD 30
→	FORWARD 30	FORWARD 30
CTRL-B	FORWARD 30	FORWARD 30
CTRL-A	FORWARD 30■	FORWARD 30
CTRL-E	FORWARD 30	FORWARD 30■
CTRL-N	FORWARD 30 RT 90	FORWARD 30 RT 90
CTRL-P	FORWARD 30 RT 90	FORWARD 30 RT 90
CTRL-O	FORWARD 30 RT 90	FORWARD 30 ■ RT 90
CTRL-D	FORWARD 30	FORARD 30
CTRL-K	FORWARD 30	F■
CTRL-Y	F■	FORWARD 30 Assuming CTRL-Y follows the CTRL-K immediately above.
←	FORWARD 30 ■ RT 90	FORWARD 30■ RT 90



Turtle Truth. Describe the function and movement of the cursor for each of the following commands when used collectively. REMEMBER NOT TO USE THE RETURN KEY AFTER EACH EDIT COMMAND WHEN WORKING IN THE EDIT MODE.

```
EDIT "TRACK
CTRL-N (3 TIMES)
CTRL-O
ST
CTRL-A
CTRL-P (2 TIMES)
CTRL-K
←
CTRL-A
CTRL-N (8 TIMES)
CTRL-O
CTRL-Y
CTRL-P
→ (12 TIMES)
CTRL-D
2
→ (7 TIMES)
CTRL-D
6
CTRL-P (7 TIMES)
CTRL-B (6 TIMES)
BIG
```

ATTENTION Your procedure should now be identical to Checkpoint 2-1. If it is not, continue editing.

```
TO BIGTRACK
PU
ST
FD 5Ø
RT 9Ø
BK 5Ø
PD
REPEAT 2 [FD 12Ø ARCR 6Ø 18Ø]
HT
END
```

Checkpoint 2-1.

```
CTRL-C
BIGTRACK
ST
PU HOME PD
TRACK
POTS
ER "BIGTRACK
POTS
```

NOTE Since BIGTRACK was developed for editing practice only, it is unnecessary to retain it for further use. ER "BIGTRACK removes BIGTRACK from the workspace.

Turtle Tester. STADIUM uses TRACK and other newly defined procedures to draw a football field with a football in the end zone. Enter the procedures, type STADIUM to observe the diagram illustrated in Figure 2-10, then perform the editing exercises and observe the gross distortions that occur with slight modifications to the existing procedures.

```
TO STADIUM
TRACK
LINES
PU BK 8Ø LT 9Ø FD 1Ø PD
SEAT
PU BK 13Ø PD
SEAT
PU FD 5Ø LT 9Ø FD 5Ø RT 9Ø PD
CROSSBAR
PU BK 35 RT 9Ø FD 135 LT 9Ø PD
CROSSBAR
PU BK 15 RT 9Ø FD 15 LT 9Ø PD
FOOTBALL
END

STADIUM

TO LINES
REPEAT 11 [RT 9Ø FD 1ØØ BK 1ØØ LT 9Ø FD 1Ø]
END

LINES
CS
STADIUM

TO SEAT
REPEAT 2 [FD 1Ø RT 9Ø FD 4Ø RT 9Ø]
END
```

```
TO CROSSBAR
BOX
FD 5 RT 90 FD 3 LT 90
FD 30
LT 90 FD 3 RT 90
BOX
END

CROSSBAR

TO BOX
REPEAT 4 [FD 5 RT 90]
END

CROSSBAR
CS
STADIUM

TO FOOTBALL
REPEAT 2 [ARCR 6 120 RT 60]
END

FOOTBALL
CS
STADIUM
```

Editing Exercises

1. In STADIUM, use CTRL-N and CTRL-K in conjunction with other commands to alter the procedure to read as follows:

```
TO STADIUM
TRACK
LINES
PU BK 80 LT 90 FD 10 PD
SEAT
PU BK 130 PD
SEAT
PU FD 50 LT 90 FD 50 RT 90 PD
CROSSBAR
FOOTBALL
END
```

2. After the procedure name LINES in STADIUM, use CTRL-K, CTRL-Y, CTRL-P and CTRL-O sequences so the procedure appears in the following form.

```
TO STADIUM
TRACK
LINES
FOOTBALL
```

```
PU BK 8Ø LT 9Ø FD 1Ø PD
SEAT
PU BK 13Ø PD
SEAT
PU FD 5Ø LT 9Ø FD 5Ø RT 9Ø PD
CROSSBAR
END
```

3. Edit the procedure LINES, using CTRL-D and other editing commands, so that the number 11 is replaced with the number 10. It should be altered to appear as follows:

```
TO LINES
REPEAT 1Ø [RT 9Ø FD 1ØØ BK 1ØØ LT 9Ø FD 1Ø]
```

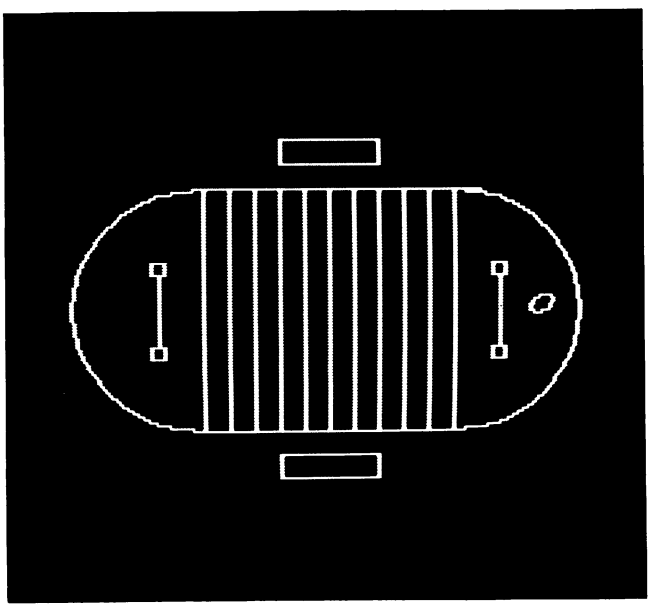
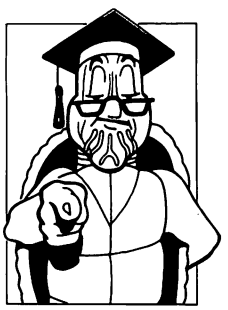


Figure 2-10.

4. In the procedure CROSSBAR, add ST immediately after the first line and insert HT just before the END command. Use CTRL-O in conjunction with appropriate editing commands so that CROSSBAR is in the following form.

```
TO CROSSBAR
ST
BOX
FD 5
RT 9Ø
FD 3Ø
LT 9Ø
FD 3
```

```
RT 90
BOX
HT
END
```



Turtle Teaser. The following procedures attempt to draw the baseball diagram illustrated in Figure 2-11. However, there are several errors to be edited. Enter the procedures into the computer. Type BASEBALL to observe the results. Use the edit mode to correct all errors indicated by the arrow (→).

```
ERPS
TO BASEBALL
HT
BOUNDARY
PU RT 135 FD 60 PD SETH 0
INFIELD
→ MOUNT
PU BK 48 LT 45 PD
BASE SETH 0
PU LT 45 FD 70 PD RT 90
→ BASE
→ PU FD 70 RT 90 PD
BASE
PU FD 70 RT 90 PD
BASE
PU FD 70 RT 90 PD
BASE SETH 00
PU LT 20 FD 15 PD SETH 0
PLAYER
PU LT 35 FD 35 PD SETH 0
PLAYER
PU RT 55 FD 50 PD SETH 0
PLAYER
PU LT 60 FD 75 SETH 0
PLAYER
PU LT 120 FD 70 PD SETH 0
PLAYER
PU RT 135 FD 45 PD SETH 0
```

```
→
    PU LT 14Ø FD 3Ø PD SETH Ø
    PLAYER
    PU RT 1Ø3 FD 46 PD SETH Ø
    PLAYER
→   PU RT 9Ø FD 65 PD SETH Ø
    PLAYER
    END

    TO BOUNDARY
→
    BK 65
    LT 45
    PD
    FD 13Ø
→
    RT 9Ø
    FD 13Ø
    LT 45
    ARCL 92 18Ø
    SETH Ø
    END

    TO INFIELD
    RT 45
    FD 7Ø
    RT 9Ø
    FD 7Ø
    LT 135
    END

    TO MOUND
    PU LT 9Ø FD 49 PD
    CIRCLER 1Ø
    SETH Ø
    END

    TO BASE
→   REPEAT 4 [FD 8 RT 85]
    END
    TO PLAYER
    CIRCLER 3
    END
```

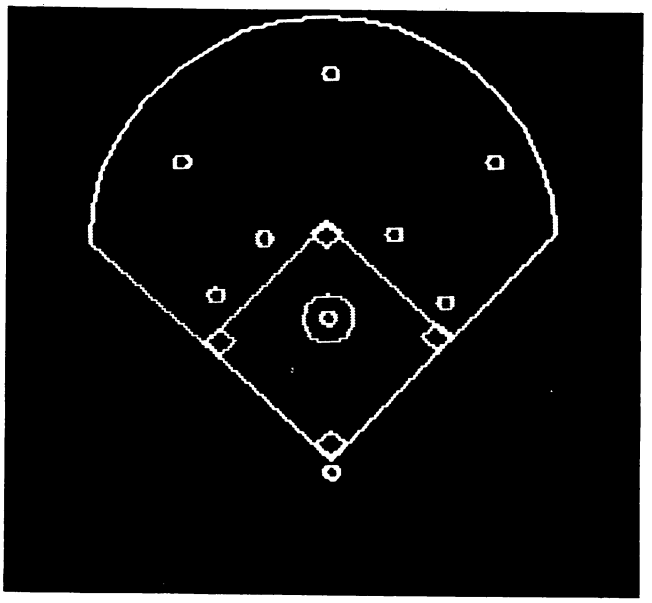
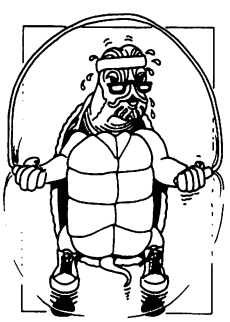


Figure 2-11.

NOTE Once the editing is completed and the procedure BASEBALL runs without any bugs, save this file as BALLPARK. Clear the workspace with ERPS.

Activity 2.5 Turtle Controls



Turtle Trials. This activity teaches the user how to add procedures to an existing file and how to correct any errors in procedure definitions.

1. Cold boot the computer with the Logo disk as usual. Insert the file diskette when instructed to do so. Type the following commands and record your observations.

   CATALOG _____

   LOAD "TURTLE.PARTS _____

   POTS _____

   SHELL _____

2. Enter the following commands and record the results.

```
TO TURTLE
SHELL
PAWS
TAIL
HEAD
EYES
PU HOME PD
END
```

3. Type in the commands and observe the screen display.

```
CS
TURTLE
FD 3Ø
RT 45
FD 11
LT 9Ø
ARCR 1Ø 18Ø
LT 9Ø
FD 11
RT 45
FD 3Ø
RT 45
FD 11
LT 9Ø
ARCR 1Ø 18Ø
```

4. Enter the following commands and record the results.

```
CTRL-T
TO PAWS
FD 3Ø
RT 45
FD 11
LT 9Ø
ARCR 1Ø 18Ø
LT 9Ø
FD 11
RT 45
FD 3Ø
RT 45
FD 11
LT 9Ø
ARCR 1Ø 18Ø
END
CS
TURTLE
```

NOTE You are now going to be entering both Logo procedure commands and CTRL key editing commands. The use of the RETURN key gets very tricky at this point. You must remember *not* to press the RETURN key after a CTRL key used for editing purposes. But you should press the RETURN key after entering a Logo procedure command. To assist your memory in this activity, we shall cue you with (n-r) following each editing command. This is to remind you not to press RETURN after using that CTRL key. If there is no (n-r), press RETURN as usual.

5. Type in the following commands and note the results.

```
EDIT "PAWS _____
CTRL-N(5 TIMES) (n-r) _____
REPEAT 4 [CTRL-E LT 90 FD 11 RT 45 FD 30 RT 45
  FD 11 LT 90]
CTRL-K (press this combination of keys until
  the cursor is on the e of the word end) (n-r)
CTRL-K (n-r) _____
CTRL-Y (n-r) _____
CTRL-C (n-r) _____
TURTLE _____
```

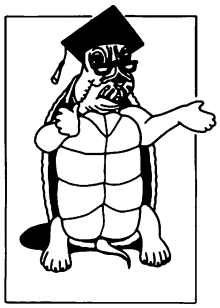
6. Enter the following commands and record the results. Commands preceded by an * indicate intentional errors. Type these commands exactly as they appear. You will correct them later.

```
CS _____
TURTLE _____
RT 90 _____
FD 31 _____
RT 45 _____
FD 11 _____
REAPEAT 3 [FD 10 LT 120] _____
HT _____
ST _____
TO TAIL _____
RT 90 _____
*FD _____
*FD 111 _____
```

*REPEAT 3 [FD 1Ø LT 12Ø] _____
END _____
CS _____
TURTLE _____
EDIT "TAIL _____
CTRL-N (n-r) _____
CTRL-N (n-r) _____
CTRL-E (n-r) _____
push the SPACEBAR once (n-r) _____
31 (n-r) _____
CTRL-A (n-r) _____
CTRL-E (n-r) _____
CTRL-A (n-r) _____
CTRL-N (n-r) _____
CTRL-O (n-r) _____
RT 45 (n-r) _____
CTRL-N (n-r) _____
CTRL-P (n-r) _____
CTRL-N (n-r) _____
CTRL-D (n-r) _____
CTRL-N (n-r) _____
CTRL-B (n-r) _____
CTRL-B (n-r) _____
← (n-r) _____
CTRL-C (n-r) _____
TURTLE _____

7. Enter the following and record the results.

CS _____
POTS _____
CTRL-T _____
SAVE "TURTLE.PARTS _____
ERASEFILE "TURTLE.PARTS _____
POTS _____
SAVE "TURTLE.PARTS _____
POTS _____
TURTLE _____
ERPS _____
TURTLE _____
CATALOG _____



Turtle Talk. In Activity 2.4 you were introduced to and practiced with the Logo editor as a means by which commands can be added, deleted, or modified. In this activity, this newly acquired skill is used during the development of the turtle illustrated in Figure 2-12. The activity emphasizes what is commonly called the top-down procedure building approach by separating the turtle into several subsidiary components; namely, a shell, four paws, a tail, and a head. In Part 2, a procedure called TURTLE defines the sequence of subprocedures by which the turtle is drawn. Part 3 demonstrates how experimenting with commands can serve as a prelude to the definition of a procedure, such as PAWS in Part 4. Part 5 uses the editor to simplify the length of PAWS. Parts 6 and 7 develop the remaining procedures for TURTLE, with the assistance of the editor to correct several deliberate errors in the procedure listings.

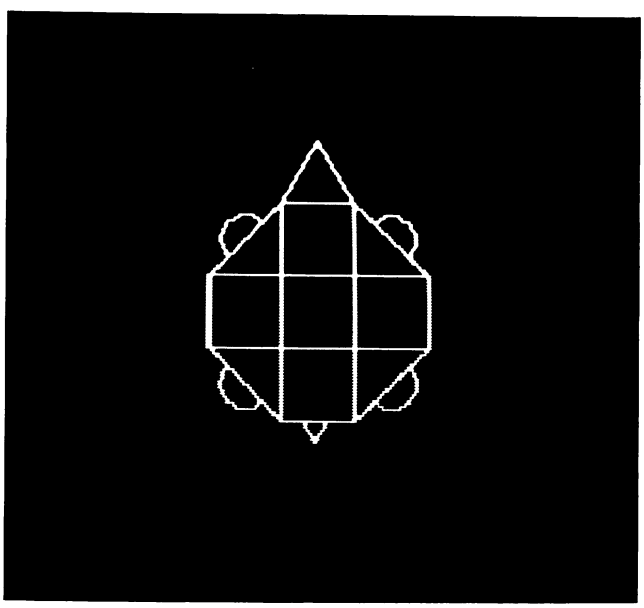
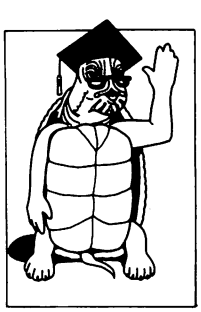


Figure 2-12.



Turtle Truth. Study the following commands and predict the result. Type them in to verify your prediction.

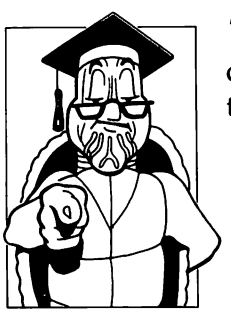
```
LOAD "TURTLE.PARTS _____
POTS _____
TURTLE _____
BK 11 _____
RT 9Ø _____
FD 9Ø _____
LT 9Ø _____
REPEAT 3 [FD 3Ø RT 12Ø] _____
CTRL-T _____
TO HEAD _____
BK 11 _____
RT 9Ø _____
FD 9Ø _____
LT 9Ø _____
REPEAT 3 [FD 3Ø RT 12Ø] _____
END _____
CS _____
TURTLE _____
EDIT "HEAD _____
CTRL-N (n-r) _____
CTRL-P (n-r) _____
CTRL-N (n-r) _____
CTRL-K (n-r) _____
CTRL-Y (n-r) _____
CTRL-A (n-r) _____
CTRL-E (n-r) _____
CTRL-B (press this combination of keys until
  the cursor is over the B) (n-r)
CTRL-O (n-r) _____
BK 1Ø (n-r) _____
use ← until the cursor erases the B (n-r)
CTRL-K (n-r) _____
CTRL-D (n-r) _____
B _____
CTRL-C (n-r) _____
TURTLE _____
ERASEFILE "TURTLE.PARTS _____
SAVE "TURTLE.PARTS _____
POTS _____
TURTLE _____
ERPS _____
```

CS _____

TURTLE _____

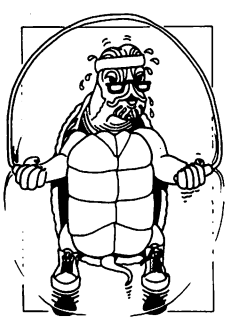
CATALOG _____

Turtle Tester. Load the file TURTLE.PARTS. Have it draw the TURTLE. Experiment and draw eyes on the turtle. Define a procedure called EYES in the definition mode but do not HT at the end. Put the procedure EYES into edit mode and move the cursor so that you can add HT on the line before END. Have it draw the TURTLE. Erase the old file called TURTLE.PARTS and save the new TURTLE.PARTS that includes the new EYES procedure. Print out all the procedures in this file. Clear the text. Print out only the procedure for TURTLE. Have it draw the turtle. Erase the old file from the workspace or computer memory. Try to print out all the titles now.



Turtle Teaser. Add the procedures to make a face on the WABBIT in the file called WABBIT.PIECES. Save the new file on the disk. Clear it from your workspace.

Activity 2.6 Turtle Maintenance



Turtle Trials. This activity reinforces the appropriate use of previously introduced CTRL characters. It also demonstrates that selected CTRL characters are available for use outside of the edit mode.

1. Cold boot the system as usual. Type the following commands and observe the results.

CATALOG _____

POTS _____

ST _____

NOTE Typing CATALOG allows you to see a listing of the files on this disk. By not loading any of these files, you are choosing to create new material. The POTS command is used in order to verify that the workspace is empty before you begin.

2. You are now going to experiment with some commands and CTRL characters in toplevel mode. In toplevel you are directing the turtle's immediate movements, rather than creating a predefined procedure to control its path.

   As you work through the following parts of this activity, you will be asked to make some intentional errors so they can be corrected by using the edit mode CTRL characters previously introduced. In Part 1, you will see that several of these CTRL characters work in toplevel, but some of them do not. The ones that do not work will result in no change of screen display and cause the computer to "beep."

   As you discover which CTRL characters are available for use at toplevel, please circle them. Notice that some are only available under certain circumstances. Please cross out the CTRL characters that do not work at toplevel.

NOTE Press RETURN after you have entered an entire line of commands unless instructed otherwise with the (n-r) indicator. The combination of CTRL keys presented on one line is to demonstrate how several keys can be used collectively to produce desired editing results.

```
HT FD 50 RT 90

CTRL-Y

CTRL-Y CTRL-A CTRL-K CTRL-Y

CTRL-Y CTRL-A CTRL-D(3 TIMES) CTRL-E

CTRL-L CTRL-T CTRL-S (n-r)

← CTRL-B CTRL-P ST CTRL-N

RT 35 CTRL-B(2 TIMES) (n-r)

← (2 TIMES) CTRL-G (n-r)

RT 45 FD 50 CTRL-A CTRL-O CTRL-E LT 45

SQUARE
```

3. You now need a procedure named SQUARE. Since it is a well known procedure, you will write this procedure in definition mode. By using definition mode, the picture that has been drawn will remain on the

screen. Please notice which CTRL characters can be used in definition mode and which ones cannot be used in this mode. Circle those that work and cross out the ones that do not work in definition mode.

```
CS
TO SQARE

CTRL-G

TO SQUARE

CTRL-N CTRL-P CTRL-O

CTRL-L CTRL-T CTRL-S (n-r)

REAPEAT 4 [FD 5Ø RT 9Ø] CTRL-A (n-r)

→ (2 TIMES) CTRL-D CTRL-E

CTRL-Y CTRL-A CTRL-K (n-r)

HT

CTRL-Y ← (2 TIMES) (n-r)

AND CTRL-B(2 TIMES) ← E (n-r)

→ (2 TIMES) CTRL-C
```

4. You are now working at toplevel again to experiment with commands necessary for drawing a CUBE. Type in the following commands and observe the results. Record you observations on the lines that are provided.

```
SQUARE _____
RT 45
FD 5Ø
LT 45
SQUARE _____
FD 5Ø
LT 135
FD 5Ø CTRL-A CTRL-K CTRL-Y _____
LT 135
FD 5Ø
LT 45
FD 5Ø
RT 135
FD 5Ø
```

```
RT 45
FD 5Ø
RT 45
FD 5Ø
RT 9Ø _____
```

5. You are now going to define the procedure CUBE, illustrated in Figure 2-13. Please notice you will start in definition mode, make an intentional mistake after doing a considerable amount of typing, and be unable to go back and correct the error while in this mode. Therefore, you will prematurely end the procedure, switch to edit mode, and then complete writing the procedure. This lets you continue without having to use CTRL-G and starting all over.

   Enter the following commands and record the results on the lines provided.

```
TO CUBE
SQUARE
RT 45
FD 5Ø
LT 45
SQARE
← CTRL-B CTRL-P END _____
  CS
CUBE _____
EDIT "CUBE _____
CTRL-N(5 TIMES) CTRL-K(3 TIMES) SQUARE
FD 5Ø
CTRL-P CTRL-K CTRL-Y _____
LT 135
CTRL-Y _____
RT 135 CTRL-A CTRL-D L _____
CTRL-E _____
CTRL-Y _____
CTRL-Y CTRL-A CTRL-O LT 45 (n-r)
CTRL-N(2 TIMES) RT 135
CTRL-Y
RT 45 CTRL-A CTRL-K CTRL-Y _____
```

NOTE Observe that the edit buffer no longer contains the previously typed line, FD 50, but now holds the most recently "killed" line, RT 45. It is now possible to "yank" back only this line.

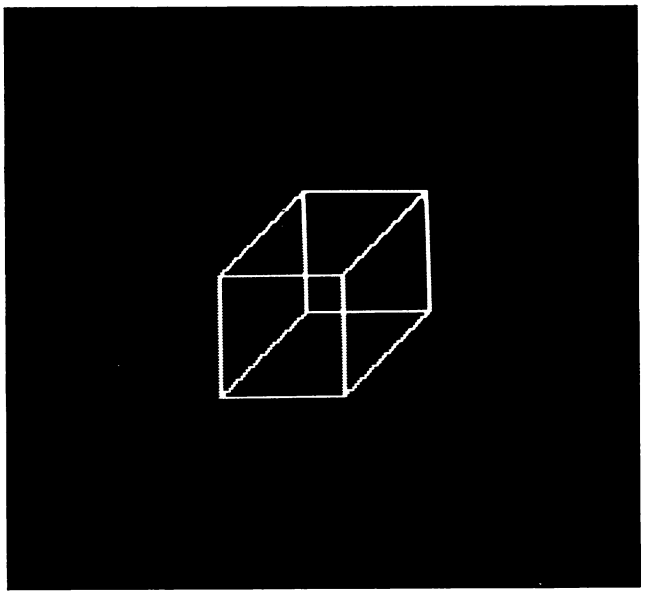


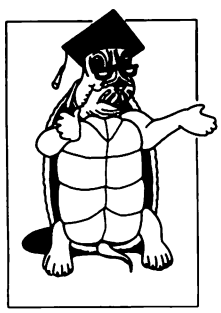
Figure 2-13.

```
TO CUBE
SQUARE
RT 45
FD 50
LT 45
SQUARE
FD 50
LT 135
FD 50
LT 135
FD 50
LT 45
FD 50
RT 135
FD 50
RT 45
FD 50
RT 45
FD 50
RT 90
END
```

Checkpoint 2-2.

```
FD 5Ø
CTRL-Y _____________________________________
FD 5Ø
RT 5Ø CTRL-B(2 TIMES) CTRL-D 9 ⟶ _____________
CTRL-L CTRL-T CTRL-S END _____________________
CTRL-C (n-r) _________________________________
CUBE _________________________________________
```

NOTE If this procedure does not resemble the picture in the illustration in Figure 2-13, compare it with Checkpoint 2-2, and make any necessary corrections to "debug" the procedure. Debug means to locate and correct errors in the procedure.



Turtle Talk. When using Apple Logo, there are three basic modes or frameworks within which you can work.

Toplevel Mode:
This mode permits you to enter primitive or user-defined commands that execute immediately. These instructions must be re-entered for each execution and do not contribute toward the development of a procedure.

Definition Mode:
This mode is implemented by typing the TO command. It permits the user to enter Logo statements which sequentially define a procedure. The procedure is named by the TO command and the following word. When entering commands in this mode, the turtle does not draw. You are able to view any previously-drawn material while you are in definition mode. The procedure is concluded by entering the word END and pressing the RETURN key. This results in a message that reports your procedure is defined. The procedure is accessed simply by typing its name each time it is to be executed.

Edit Mode:
This mode is used to modify procedures. It is accessed by typing EDIT "procedure name (for example, EDIT "CUBE). The screen display changes to show the procedure that you have chosen to edit and

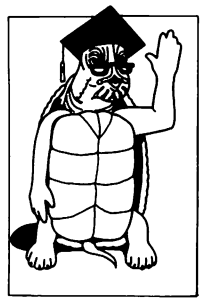
the bottom of the screen informs you that you are in the Logo editor. All of the editing CTRL keys are available for use in modifying the procedure. To leave this mode you must type CTRL-C and press RETURN. You will receive a message indicating the procedure is defined. You cannot view any drawing while in this mode.

Activity 2.6 reviews the various Logo control commands introduced in Activities 2.4 and 2.5, and allows you to experiment with their use in three different Logo modes. You should have discovered that all CTRL commands are not available in every mode. Table 2-2 provides a summary of the availability of the control commands in each of the three operation modes.

Table 2-2. Summary of control commands available in three operation modes.

Keys	Toplevel	Definition	Edit
	Horizontal Control		
←	Y	Y	Y
→	N*	N*	N
CTRL-B	Y	Y	Y
CTRL-A	Y	Y	Y
CTRL-E	Y	Y	Y
	Vertical Control		
CTRL-N	N	N	Y
CTRL-P	N	N	Y
CTRL-O	N	N	Y
	Text Manipulation		
CTRL-D	Y	Y	Y
CTRL-K	Y	Y	Y
CTRL-Y	Y**	Y	Y
	Screen Display		
CTRL-L	Y	Y	N
CTRL-T	Y	Y	N
CTRL-S	Y	Y	N
	Level Control		
CTRL-G	Y	Y	N
CTRL-C	N	N	Y

*This is true if this command follows a CTRL-B or a CTRL-A.
**In this mode, CTRL-Y yanks back the previously-typed line or the most recently killed sequence of characters.



Turtle Truth. Examine the following commands and predict the results. Enter the commands to check your predictions. Compare your screen display to the illustrations in Figure 2-14.

```
CS
CUBE RT 9Ø CTRL-A CTRL-K CTRL-Y
CTRL-Y
CTRL-Y
CTRL-Y
CTRL-L (n-r)
CTRL-T (n-r)
TO 4CUBE
REPEAT 4 [CUBE RT 9Ø]
HT
END
CTRL-S (n-r)
CS
4CUBE
EDIT "4CUBE
→(3 TIMES) CTRL-D 6 CTRL-N (n-r)
CTRL-A CTRL-O FULLSCREEN
CTRL-K →(7 TIMES) CTRL-D (n-r)
6 CTRL-E CTRL-B(3 TIMES) CTRL-D 6 (n-r)
CTRL-C (n-r)
6CUBE
```

NOTE If this 6CUBE procedure does not resemble the illustration in Figure 2-14, compare it with Checkpoint 2-3 and make any necessary corrections to debug it.

```
TO 6CUBE
FULLSCREEN
REPEAT 6 [CUBE RT 6Ø]
HT
END
```

Checkpoint 2-3.

```
CTRL-T
POTS
CTRL-S
```

NOTE The 4CUBE procedure still exists. By creating 6CUBE, you have merely modified the original procedure and defined it in another form.

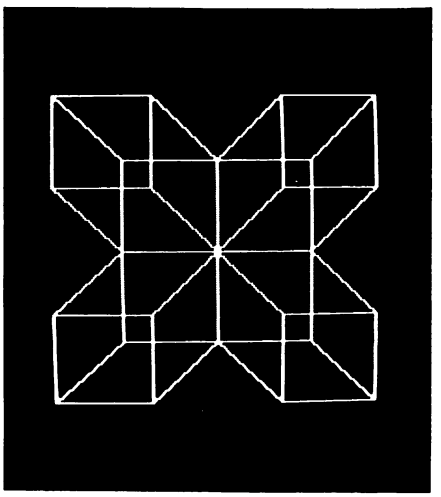


Figure 2-14a.

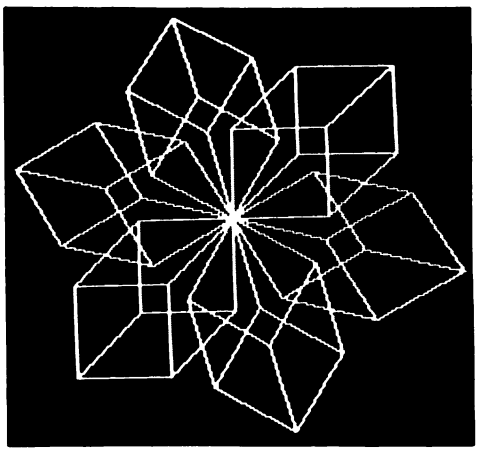
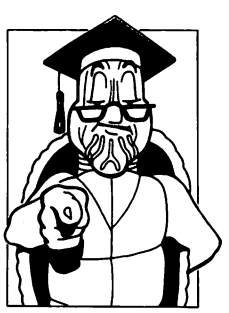


Figure 2-14b.

Turtle Tester. Edit the procedure 6CUBE and define a new procedure named 8CUBE which is illustrated in Figure 2-15 and reads as follows:

```
TO 8CUBE
FULLSCREEN
REPEAT 8 [CUBE LT 45]
HT
END
```



Turtle Teaser. Edit one of the previous procedures to draw the design illustrated in Figure 2-16.

NOTE Save all the procedures created in this activity under the file name 3.D.DESIGNS.

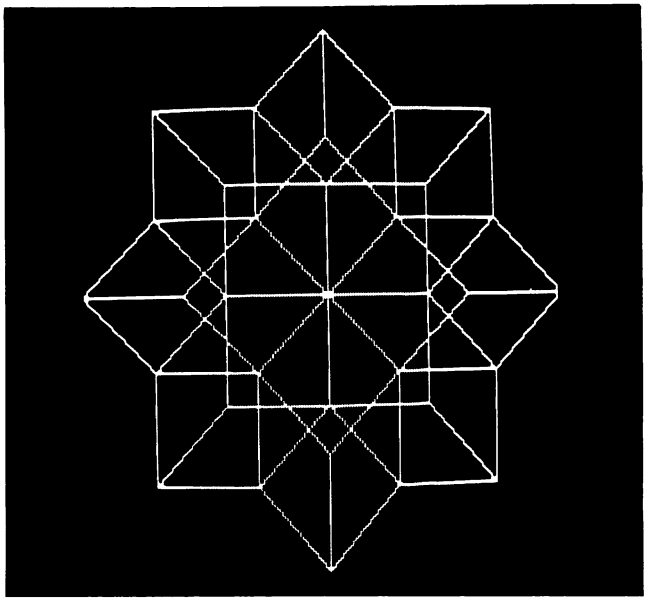


Figure 2-15.

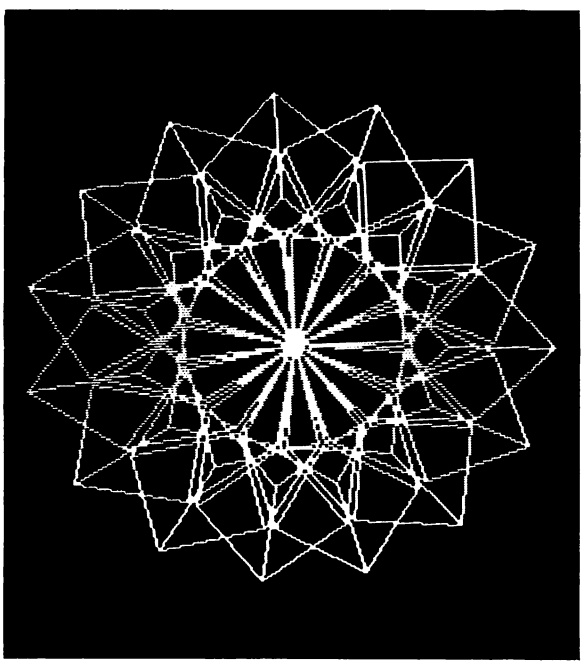


Figure 2-16.

Reference Point 2

Command Review. Table 2-3 contains all of the CTRL keys introduced in the chapter. Each character is named, its function described, and its availability in each mode is indicated by an *. [TL stands for toplevel mode, DEF stands for definition mode, and EDIT stands for edit mode.)

Table 2-3. CTRL keys introduced in Chapter 2.

CTRL-Key	*Function*	*TL*	*DEF*	*EDIT*
CTRL-A	Moves ■ to beginning of the line	*	*	*
CTRL-B	Moves ■ one space back without erasing	*	*	*
CTRL-C	Completes an edited procedure and adds procedure to workspace			*
CTRL-D	Deletes character directly under the ■	*	*	*
CTRL-E	Moves ■ to end of current line	*	*	*
CTRL-K	Erases everything on current line to the right of ■	*	*	*
CTRL-L	Scrolls the screen up to place current line in center of screen display			*
CTRL-L	Devotes whole screen to graphics display	*	*	
CTRL-N	Moves ■ to next line of a procedure at current ■ position			*
CTRL-O	Opens a space for a new line directly above or before ■			*
CTRL-P	Moves ■ to previous line			*
CTRL-S	Splits the screen—top for graphics—bottom 4 lines for text	*	*	
CTRL-T	Devotes entire screen to text	*	*	
CTRL-W	Stops the running of Logo until any other character is typed	*	*	

Table 2-3. CTRL keys introduced in Chapter 2 *(continued).*

CTRL-Key	*Function*	*TL*	*DEF*	*EDIT*
CTRL-Y	Inserts the line that was last "killed" by CTRL-K or that was previously typed	*	*	*
CTRL-Z	Stops running of Logo making it pause until CO or CONTINUE is typed		*	
←	Moves ■ back one space and erases characters it moves over	*	*	*
→	Moves ■ ahead one space	*	*	*

Application. Enter the error-filled procedures listed below and illustrated in Figure 2-17a. Correct the procedures, using editing commands to produce the correct version also printed below and illustrated in Figure 2-17b. When the procedures work correctly, save the procedures in a file named PICTURE.

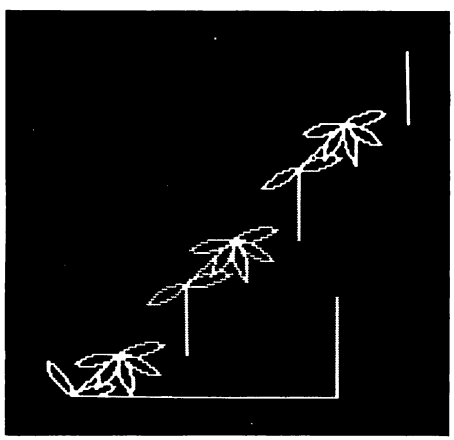


Figure 2-17a.

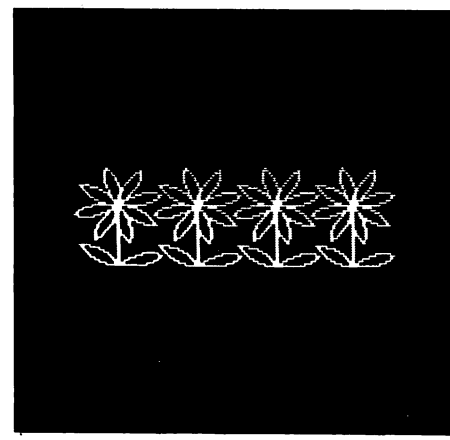


Figure 2-17b.

ERROR FILLED

```
TO GARDEN
BK 5Ø
LT 9Ø
FD 12Ø
RT 9Ø
PD
REPEAT 3 [FLOWER RT 9Ø
  FD 35 LT 9Ø]
END


TO FLOWER
LT 6Ø
PETAL
SETH Ø
RT 45
PETAL
FD 3Ø
REPEAT 5 [PETAL
  RT 36Ø/8]
PU
BK 3Ø
PD
END


TO PETAL
ARCR 2Ø 6Ø
RT 12Ø
ARCR 2Ø 6Ø
RT 12Ø
END
```

CORRECT VERSION

```
TO GARDEN
PU
BK 5Ø
LT 9Ø
FD 12Ø
RT 9Ø
PD
REPEAT 4 [FLOWER
  PU RT 9Ø FD 35
  LT 9Ø PD]
PU HOME PD
END


TO FLOWER
LT 9Ø
PETAL
SETH Ø
RT 45
PETAL
SETH Ø
FD 3Ø
REPEAT 8 [PETAL
  RT 36Ø/8]
PU
BK 3Ø
PD
END


TO PETAL
REPEAT 2 [ARCR
  2Ø 6Ø RT 12Ø]
END
```

Suggested Projects

Go back into some of the previous files and alter the procedures using your new skill at editing. Save the new procedures you create under a different file name so that you still have the original file to use in future activities. For example:

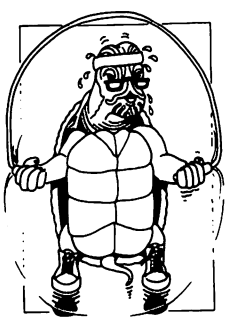
REGULAR.POLYGONS Change the length of the sides to make the shapes larger or smaller.

WABBIT.PIECES Change the formation of the facial features you added to the rabbit face.

3

Extending Turtle Trips

Activity 3.1 Turtle Input



Turtle Trials. This activity introduces the user to the design of procedures that include one input variable.

1. Enter the following commands and describe the screen display on the line provided.

```
ST
TO SQUARE
REPEAT 4 [FD 5Ø RT 9Ø]
END
SQUARE
```

_____

2. Use the editor to modify the SQUARE procedure to read as follows: (edit using the command EDIT "SQUARE and conclude the editing with CTRL-C):

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 9Ø]
END
```

```
SQUARE _____
TO SQUARE :SIDE _____
SQUARE :1Ø _____
SQUARE 1Ø _____
SQUARE 3Ø _____
SQUARE 5Ø _____
SQUARE 7Ø _____
```

3. Type in the following commands and record your observations where indicated.

```
CTRL-T
TO SQUARES _____
SQUARE 1Ø
SQUARE 3Ø
SQUARE 5Ø
SQUARE 7Ø
HT
END _____

CTRL-S
CS
SQUARES _____
```

4. Enter the commands as shown and record your observations on the lines. Compare your screen displays to Figures 3-1 and 3-2.

```
CS
ST
REPEAT 4 [SQUARES RT 36Ø/4]
TO DESIGN :MANY
REPEAT :MANY [SQUARES RT 36Ø/:MANY]
END

CS
ST
DESIGN _____
DESIGN 4 _____
CS
DESIGN 8 _____
CTRL-L
CTRL-S
```

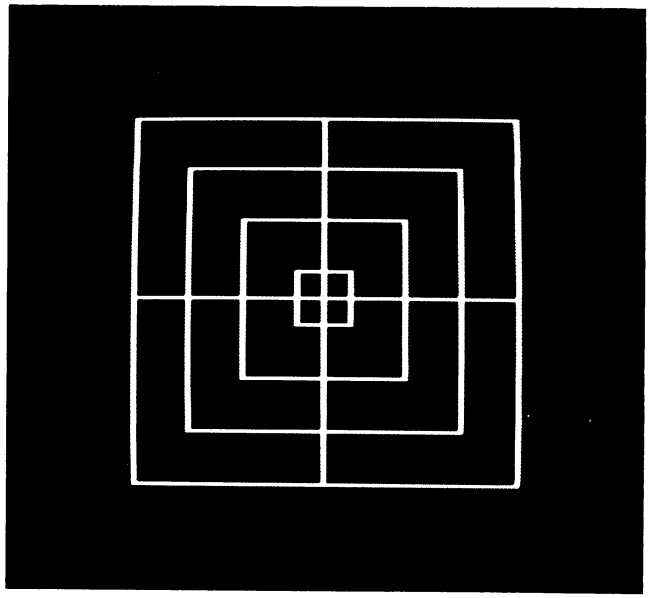


Figure 3-1.

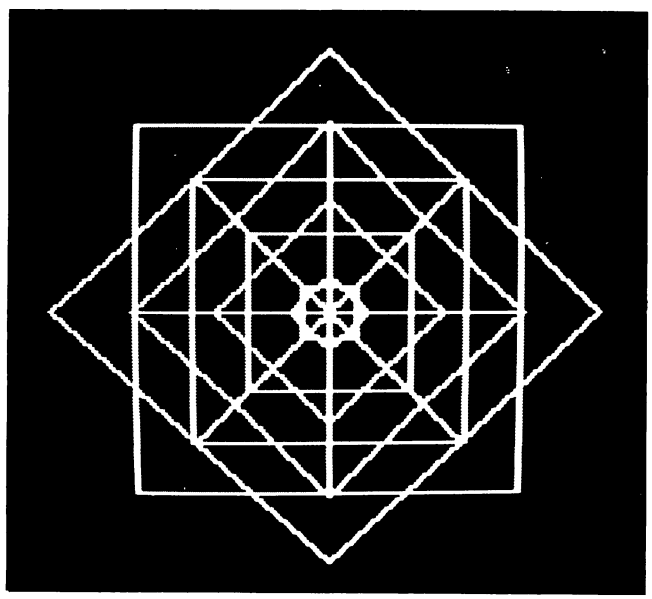


Figure 3-2.

5. Type in the following commands and describe what you see. Compare your screen displays with Figures 3-3, 3-4, and 3-5.

```
CS
ST
```

```
DESIGN 5 ____________________________________
CTRL-L
CTRL-S
CS
ST
DESIGN 9 ____________________________________
CTRL-L
CTRL-S
CS
DESIGN 2 ____________________________________
```

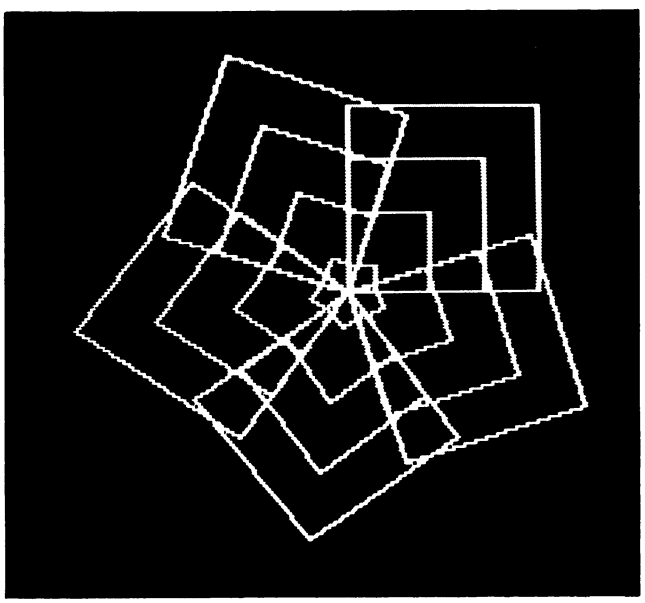


Figure 3-3.

6. Use the editor to modify the SQUARE and DESIGN procedures to read as follows:

```
TO SQUARE :S
REPEAT 4 [FD :S RT 90]
END

TO DESIGN :M
REPEAT :M [SQUARES RT 360/:M]
END

POPS
```

NOTE Use the screen display following POPS to verify your editing.

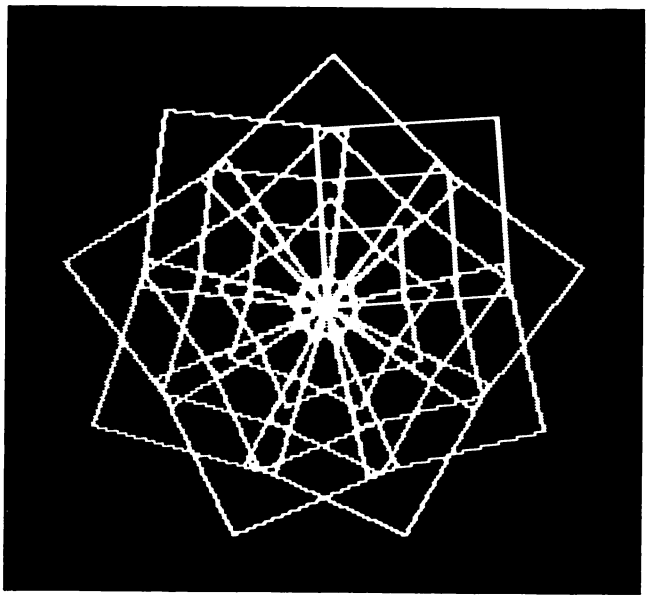


Figure 3-4.

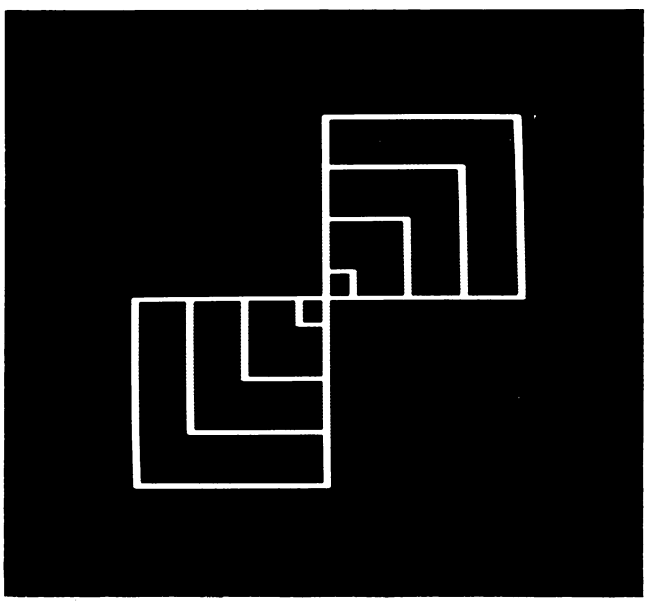


Figure 3-5.

7. Enter the following commands and observe the results. Figure 3-6 displays DESIGN 12.

```
DESIGN 12
CTRL-L
```

```
CS
CTRL-S
ST
DESIGN M (Enter your own number for the input variable M.)
```

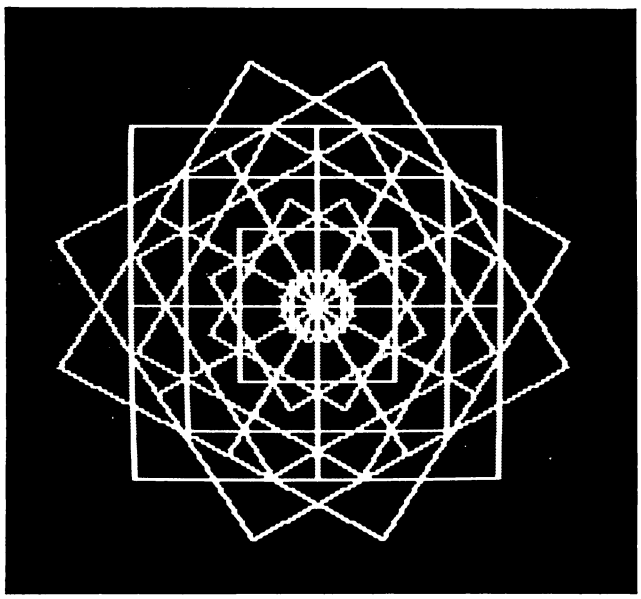
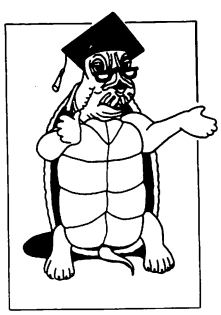


Figure 3-6.



Turtle Talk. One of the more powerful aspects of any programming language is the ability to use input variables. Part 1 introduces the concept of an input variable by contrasting two procedures:

```
TO SQUARE
REPEAT 4 [FD 5Ø RT 9Ø]
END
```

and

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 9Ø]
END
```

The previous activities have not used input variables. Instead, all commands have included constants. Constants are values which never

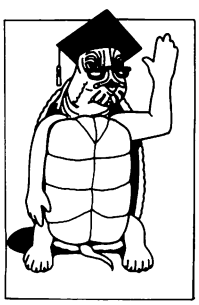
change. SQUARE contains three constants: 4, 50, and 90. SQUARE is restricted to drawing a square with each side having a length of 50 units.

In contrast to a constant, an input variable can be considered to be a memory location which is identified by a name or group of characters (such as SIDE), and holds a value that can change. An input variable can be assigned any number. The procedure SQUARE :SIDE permits the square to be a size determined by providing SIDE with a numerical value. For example, SQUARE 10 draws a square with the length of each side being 10 units. SQUARE 70 draws a square with the length of each side being 70 units. Variables provide the programmer with much more versatility when defining procedures.

Part 4 defines a procedure that rotates square patterns. The DESIGN procedure includes a variable by the name of MANY to control the number of rotations. The expression REPEAT :MANY [SQUARE RT 360/ :MANY] assures the completion of the TOTAL TURTLE TRIP. By including MANY in DESIGN, a variety of square rotational patterns can be displayed simply by entering commands such as DESIGN 4, DESIGN 9, or DESIGN N, where N is a value provided by the user.

Part 5 demonstrates that the name selected for the variable is not critical. The replacement of S for SIDE and M for MANY does not alter the designs produced by the procedures. One-letter variable names are more convenient for the less skilled typist, but are not as meaningful for other readers (or the program author at a later time) and are to be discouraged. Full mnemonic names are used throughout the remainder of the book. The reader may substitute abbreviations if so desired.

Activity 3.1 introduces you to the use of one variable when defining procedures. Future activities will reinforce and extend your ability to incorporate more than one variable into procedures.



Turtle Truth. Study the following procedure and project its outcome. Enter the commands and check your prediction. Compare the screen display to the pictures illustrated in Figures 3-7, 3-8, 3-9, and 3-10.

```
CS
TO TRIANGLE :SIDE
REPEAT 3 [FD :SIDE RT 12Ø]
END

TRIANGLE 4Ø
```

```
TRIANGLE 6Ø
TRIANGLE 8Ø
TRIANGLE 9Ø

TO TRIANGLES
TRIANGLE 4Ø
TRIANGLE 6Ø
TRIANGLE 8Ø
TRIANGLE 9Ø
END

TO TRIDESIGN :MANY
FULLSCREEN
REPEAT :MANY [TRIANGLES RT 36Ø/:MANY]
END

CS
TRIDESIGN 2
CTRL-S
CS
TRIDESIGN 6
```

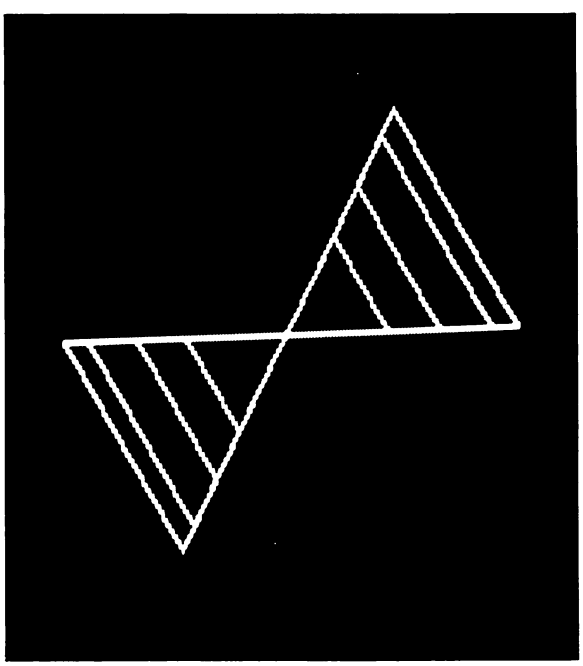


Figure 3-7.

```
CTRL-S
CS
TRIDESIGN 9 ____________________________
CTRL-S
CS
TRIDESIGN 17 ___________________________
CTRL-S
CS
```

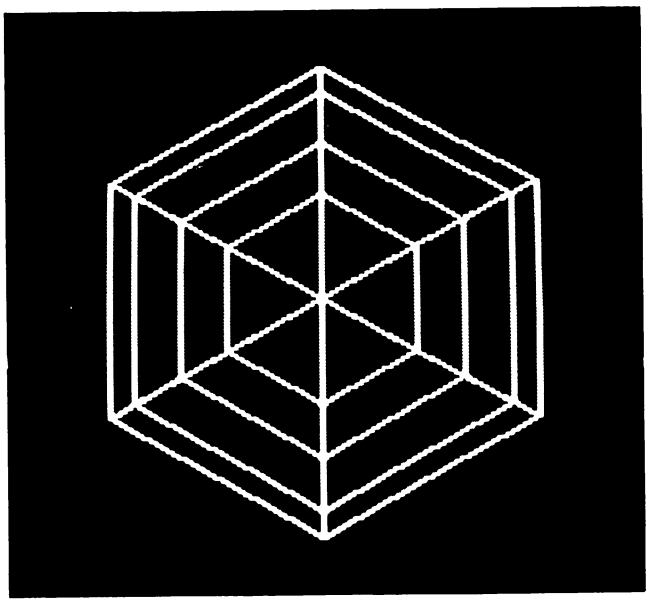


Figure 3-8.

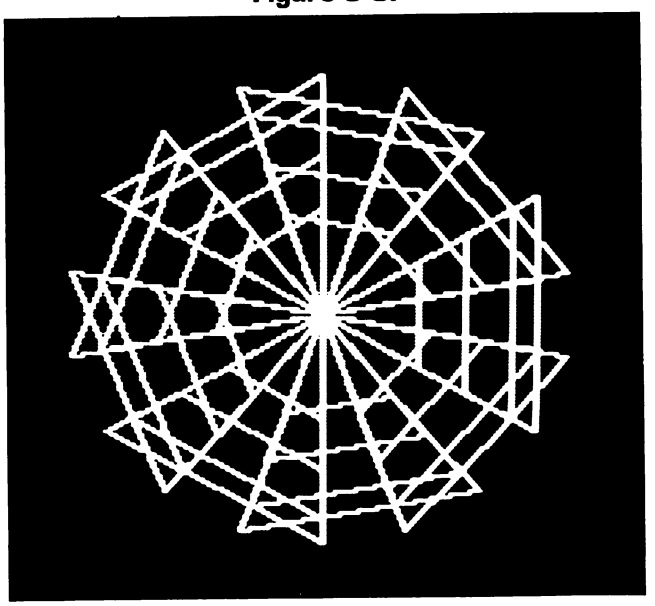


Figure 3-9.

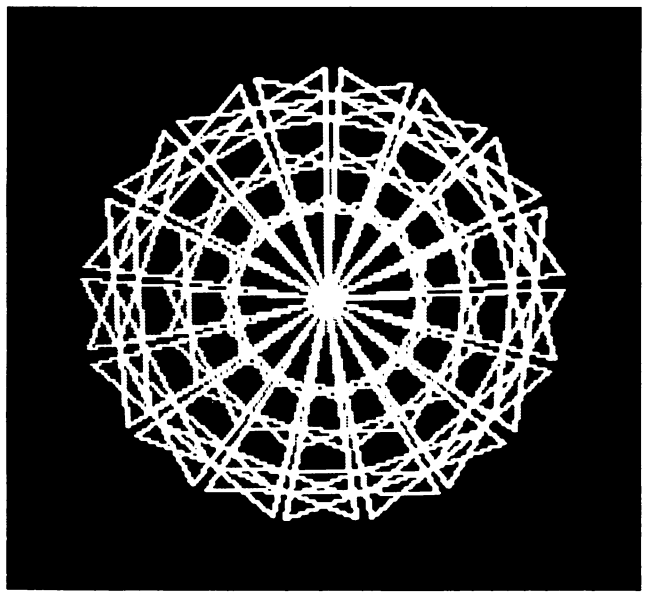
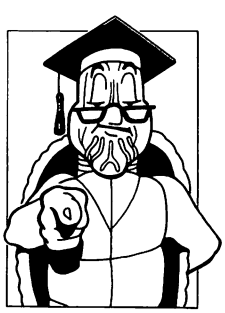


Figure 3-10.

NOTE Save these procedures under the file name VAR.ROTATIONS.

Turtle Tester. Erase the procedure in the workspace with ERPS. Define the new procedures suggested below and save them in a file named VAR.ROTATIONS2.

1. Define a procedure called SQUARES.2. In the procedure, include different inputs for the square variable SIDE.
2. Define a procedure named SQDESIGN.2 which uses the variable MANY and the newly defined procedure SQUARES.2. Experiment using different inputs for MANY and observe the resulting changes in the displays. (Answer on page 231.)



Turtle Teaser. Erase the workspace. Loan the file named VAR.ROTATIONS. Study the illustration in Figure 3-11. Using the existing procedures, define a new procedure named COMBO that duplicates the illustration. When this is accomplished, type the following to add COMBO to your VAR.ROTATIONS file:

```
ERASEFILE "VAR.ROTATIONS
SAVE "VAR.ROTATIONS
```

NOTE Apple Logo does not allow you to SAVE a file by the same name as one that already exists on diskette. If you wish to update or replace a file on diskette, as with VAR.ROTATIONS, you must remove the current file on diskette with the ERASEFILE command, then SAVE the modified version from workspace.

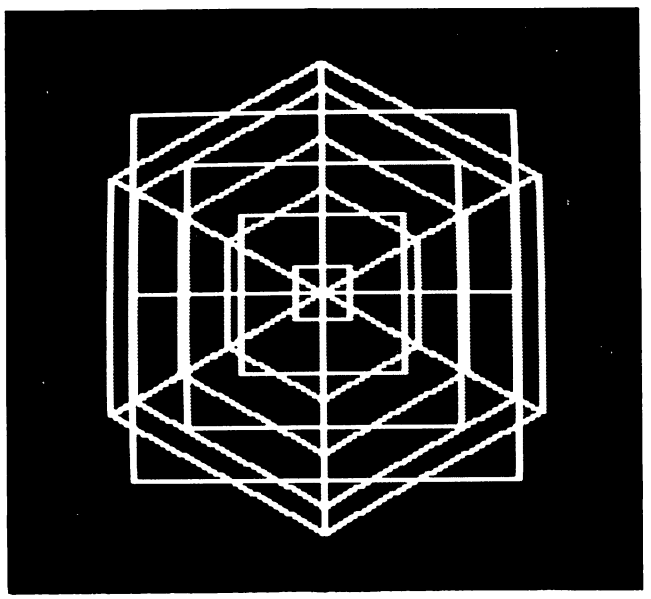
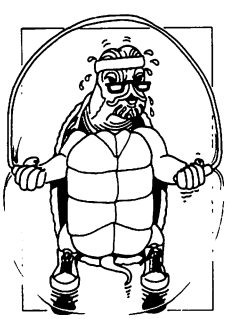


Figure 3-11.

Activity 3.2 Turtle Hues



Turtle Trials. This activity uses color to introduce the user to the design of procedures that include more than one input variable.

1. Enter the following commands and describe the screen display on the lines provided.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

SQUARE _____
SQUARE 20 _____
CS
SQUARE 40 _____
```

2. Use the editor to modify the SQUARE procedure to read as follows (edit using the command, EDIT "SQUARE):

```
TO SQUARE :SIDE
SETPC 2
REPEAT 4 [FD :SIDE RT 90]
END

SQUARE 30 _____
```

3. Use the editor to alter the SQUARE procedure to read as follows:

```
TO SQUARE :SIDE :COLOR
SETPC :COLOR
REPEAT 4 [FD :SIDE RT 90]
END

SQUARE 40 _____
SQUARE 40 :3 _____
SQUARE 40 3 _____
SQUARE 20 5 _____
```

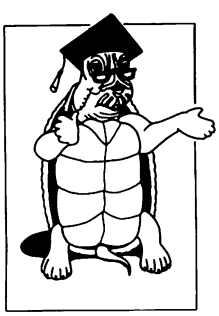
4. Enter the commands and describe the screen display on the lines provided.

```
CS
TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END

RECTANGLE 10 40 _____
SETPC 1
RECTANGLE 10 40 _____
CS
```

```
RECTANGLE 3Ø 3Ø _____________________________
CS
SETPC 5
RECTANGLE 5 6Ø _____________________________
```

5. Use the editor to modify the RECTANGLE procedure to read as follows:

```
TO RECTANGLE :LENGTH :WIDTH :COLOR
SETPC :COLOR
REPEAT 2 [FD :LENGTH RT 9Ø FD :WIDTH RT 9Ø]
END

RECTANGLE 2Ø 4Ø _____________________________
RECTANGLE 2Ø 4Ø 4 ___________________________
CS
RECTANGLE 6Ø 1Ø 5 ___________________________
CS
RECTANGLE 4Ø 4Ø 1 ___________________________
CS
RECTANGLE 8Ø 5Ø 2 ___________________________
CS
RECTANGLE 1ØØ 8Ø 3 __________________________
CS

SAVE "COLOR.SHAPES
ERPS
```



Turtle Talk. Procedures can be defined to include more than one input variable. In Part 1, the variable SIDE is used in the procedure SQUARE to define the length of each side. Color is introduced in Part 2 with the SETPC (SETPenColor) command. SETPC N, where N = 0, 1, 2, 3, 4, or 5 designates the color for the diagram to be drawn. Table 3-1 displays the color each number represents when drawn on a black background screen.

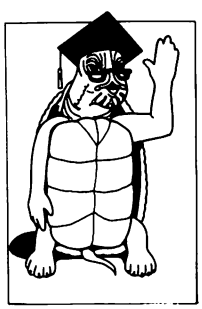
SQUARE is modified in Part 3 to include two variables (SIDE representing the length of each side, and COLOR designating the drawing color). Now SQUARE may be drawn to any size and in any color. Simply enter SQUARE followed by two input variables, separated by a space.

The use of these variables in the definition of a procedure is illustrated in Part 5. The various sizes and colors of rectangles drawn in Part 5 demon-

Table 3-1. Color representations on a black screen.

Color	*Number*
black	0
white	1
green	2
violet	3
orange	4
blue	5

strate the possible selection of hues available in Logo. The versatility and ease of using variables to define procedures is evident.



Turtle Truth. Study the following commands and predict the results. Enter the commands to confirm your prediction. (The file PICTURE was created in the REFERENCE POINT for Chapter 2.)

```
LOAD "PICTURE

TO STEM :COLORSTEM
SETPC :COLORSTEM
LT 90
LEAF
RT 90
LEAF
SETH 0
FD 30
END

TO LEAF
REPEAT 2 [ARCR 23 60 RT 120]
END
```

Use the editor to modify the FLOWER and GARDEN procedures to read as follows:

```
TO FLOWER :COLORFLOWER
SETPC :COLORFLOWER
REPEAT 8 [PETAL RT 360/8]
PU
BK 30
PD
END
```

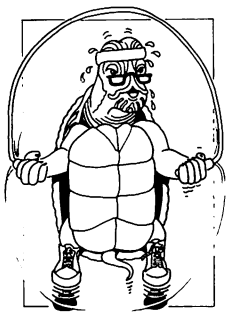
```
TO GARDEN :COLORSTEM :COLORFLOWER
PU
BK 5Ø
LT 9Ø
FD 12Ø
RT 9Ø
PD
REPEAT 4 [STEM :COLORSTEM FLOWER :COLORFLOWER PU RT 9Ø FD 35 LT
  9Ø PD]
PU HOME PD
END

GARDEN 2 3 _____
CS
GARDEN 5 4 _____
CS
GARDEN 2 1 _____
```

NOTE Save these procedures in a file named PICTURE2.

Turtle Tester. Write a program that uses two variables and creates a different flower shape in several colors. Save these procedures under a name of your choice.

Turtle Teaser. Write a program that draws a row of flowers. Alternate the drawing of the flower created in Turtle Truth with the one created in Turtle Tester.

Activity 3.3 Turtle Background



Turtle Trials. This activity demonstrates the possibilities for drawing on a non-black background screen, and introduces a method for displaying messages on the screen.

1. Enter the following commands and describe the screen display on the lines provided.

```
SETBG 1 _____
SETBG 2 _____
SETBG 3 _____
SETBG 4 _____
SETBG 5 _____
SETBG 6 _____
SETBG 7 _____
SETBG 1 _____
SETBG Ø _____

TO FLASH
SETBG 1
SETBG 2
SETBG 3
SETBG 4
SETBG 5
SETBG Ø
END

FLASH _____
```

2. Use the editor to modify the FLASH procedure to read as follows:

```
TO FLASH2 :TIME
SETBG 1
WAIT :TIME
SETBG 2
WAIT :TIME
```

```
SETBG 3
WAIT :TIME
SETBG 4
WAIT :TIME
SETBG 5
WAIT :TIME
SETBG Ø
WAIT :TIME
END
```

FLASH2 1Ø _____

FLASH2 5Ø _____

FLASH2 1ØØ _____

3. Modify the FLASH2 procedure to read as follows and enter the new MESSAGE procedure. Observe and record the results of the screen displays.

```
TO FLASH3 :TIME
CLEARTEXT
SETBG 1
MESSAGE :TIME
SETBG 2
MESSAGE :TIME
SETBG 3
MESSAGE :TIME
SETBG 4
MESSAGE :TIME
SETBG 5
MESSAGE :TIME
SETBG 6
MESSAGE :TIME
PRINT [SETBG 6 IS THE BLACK TO USE WITH]
PRINT [A B & W MONITOR]
WAIT :TIME + 1ØØ
CLEARTEXT
SETBG Ø
MESSAGE :TIME
PRINT [Ø IS ALSO BLACK]
PRINT [ ]
PRINT [SETBG Ø IS THE DEFAULT COLOR]
PRINT [FOR THE SCREEN]
```

```
WAIT :TIME + 1ØØ
CLEARTEXT
END

TO MESSAGE :TIME
PRINT [THE BACKGROUND COLOR IS]
PRINT BG
WAIT :TIME
CLEARTEXT
END
```

FLASH3 1ØØ _____

FLASH3 1Ø _____

FLASH3 2ØØ _____

4. Enter the following commands and observe the results. Record your observations on the lines provided.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 9Ø]
END

TO TRIANGLE :SIDE
REPEAT 3 [FD :SIDE LT 12Ø]
END

TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH LT 9Ø FD :WIDTH LT 9Ø]
END

TO HOUSE :SIDE
ST
SQUARE :SIDE
FD :SIDE RT 9Ø
TRIANGLE :SIDE
FD :SIDE/2
PU RT 9Ø FD :SIDE/2 PD
RECTANGLE :SIDE/2 :SIDE/3
HT
END

CS
```

```
SETBG 1 _____
SETPC 2 _____
HOUSE 5Ø _____
CS
SETBG 2 _____
SETPC 3 _____
HOUSE  6Ø _____
CS
SETBG 1 _____
SETPC Ø
PRINT PC _____
PR BG _____
HOUSE 2Ø _____
```

5. Enter the following commands and observe the effect of each PenColor as it draws on each BackGround color. Describe the color of the lines in each drawing.

```
TO HOUSEDEMO :SIDE :TIME
HOUSE :SIDE
WAIT :TIME
CS
END

TO COLORDEMO :COLOR :SIDE :TIME
CLEARTEXT
SETPC :COLOR
SETBG 1
PRINT [PENCOLOR IS ] PRINT :COLOR
HOUSEDEMO :SIDE :TIME
SETBG 2
HOUSEDEMO :SIDE :TIME
SETBG 3
HOUSEDEMO :SIDE :TIME
SETBG 4
HOUSEDEMO :SIDE :TIME
SETBG 5
HOUSEDEMO :SIDE :TIME
SETBG Ø
HOUSEDEMO :SIDE :TIME
CLEARTEXT
END
```

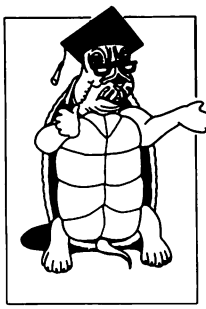
```
COLORDEMO 1 65 2ØØ
  white PC on white BG _____
            on green BG _____
            on violet BG _____
            on orange BG _____
            on blue BG _____
            on black BG _____

COLORDEMO 2 2Ø 15Ø
  green PC on white BG _____
            on green BG _____
            on violet BG _____
            on orange BG _____
            on blue BG _____
            on black BG _____

COLORDEMO 3 3Ø 2ØØ
  violet PC on white BG _____
             on green BG _____
             on violet BG _____
             on orange BG _____
             on blue BG _____
             on black BG _____

COLORDEMO 4 4Ø 15Ø
  orange PC on white BG _____
             on green BG _____
             on violet BG _____
             on orange BG _____
             on blue BG _____
             on black BG _____

COLORDEMO 5 5Ø 2ØØ
  blue PC on white BG _____
           on green BG _____
           on violet BG _____
           on orange BG _____
           on blue BG _____
           on black BG _____
```

```
COLORDEMO Ø 6Ø 15Ø
```

```
black PC on white BG _____
         on green BG _____
         on violet BG _____
         on orange BG _____
         on blue BG _____
         on black BG _____
```

6. Modify the MESSAGE procedure and enter the two new procedures. Then type in the commands and observe the results. Notice that an object first drawn in color on a black screen changes peculiarly as the background color is altered.

```
TO MESSAGE2
PRINT [THE PEN COLOR IS]
PRINT PC
END

TO BGFLASH :SIDE :TIME
HOUSE :SIDE
MESSAGE2
WAIT :TIME
FLASH2 :TIME
CLEARTEXT
CS
END

TO FLASHDEMO :SIDE :TIME
SETPC 1
BGFLASH :SIDE :TIME
SETPC 2
BGFLASH :SIDE :TIME
SETPC 3
BGFLASH :SIDE :TIME
SETPC 4
BGFLASH :SIDE :TIME
SETPC 5
BGFLASH :SIDE :TIME
SETPC Ø
BGFLASH :SIDE :TIME
END

FLASHDEMO 6Ø 2ØØ
```



Turtle Talk. Logo allows for the drawing of diagrams on a colored background screen by using the SETBG (Set BackGround) command, followed by a numerical input from 0 to 6. The screen can be colored white (1), green (2), violet (3), orange (4), blue (5), or black (0 or 6). The input 6 is best used with a black and white monitor because the lines are drawn more finely and appear more distinctly.

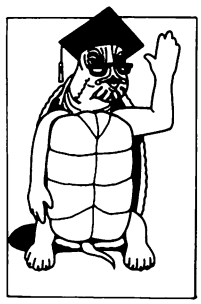
In Part 2, the WAIT command is used to hold a screen display and cause a pause in the running of a procedure. The command is written WAIT N, where N is an input number. Each unit is equivalent to 1/60th of a second. Part 2 uses the WAIT command with a variable TIME. TIME is given several values during the execution of the activity.

It is possible to display text and a graphics screen by using the PRINT or PR command. The four bottom lines are available for text display, although the fourth line usually contains the cursor. The message is displayed by entering PRINT followed by the text enclosed within the square brackets ([ ]) The line PRINT [ ] is used to create a space between printed messages to improve the appearance of the display. It is also possible to print the output of a built-in Logo procedure such as BG to receive the current background color (similar to the previously discussed PR HEADING command). The printing of BG during the running of a procedure (as demonstrated in Part 3) allows the viewer to observe the changing value of BG.

Notice in Parts 1, 2, and 3, the procedure FLASH is modified several times. The originals are left intact so they may be used in their initial forms. Each modification is assigned a different name by appending a number, then is newly defined by typing CTRL-C.

Part 4 involves a sophisticated use of variables within a procedure. Each sub-procedure is assigned a variable in proportion to the controlling variable entered when calling the main procedure. In this way, the procedure HOUSE :SIDE is able to draw a house of any size with all parts in a consistent proportion.

Part 5 demonstrates various sized houses drawn in each PenColor on a variety of background colors. These demos display the limitations of drawing colored lines on different colored backgrounds. Your observations and notes during these parts should indicate that black and white pens draw on any background other than themselves and any color pen draws on a black or white background. However, there are some peculiar results with other combinations. On an orange or blue background, green lines appear orange and violet lines appear blue. Conversely, on violet or green backgrounds, orange lines appear green and blue lines appear violet. Even stranger displays occur if you change the background color after the lines have already been drawn. It should be mentioned that these strange effects occur because of the Apple computer system and not because of Logo.



Turtle Truth. Examine the following commands and predict the results. Enter the commands to check your predictions and compare the final screen display with Figure 3-12.

```
TO PANES :SIDE
SQUARE :SIDE/4
SQUARE :SIDE/8
FD :SIDE/8 RT 9Ø
FD :SIDE/8 LT 9Ø
SQUARE :SIDE/8
LT 9Ø FD :SIDE/8
RT 9Ø BK :SIDE/8
END
```

Modify the HOUSE procedure to read as follows:

```
TO HOUSE2 :SIDE :COLORPANES :COLORSQUARE :COLORTRIANGLE
  :COLORRECTANGLE
ST
SETPC :COLORSQUARE
SQUARE :SIDE
FD :SIDE RT 9Ø
SETPC :COLORTRIANGLE
TRIANGLE :SIDE
FD :SIDE/2 PU RT 9Ø FD :SIDE/2 PD
SETPC :COLORRECTANGLE
RECTANGLE :SIDE/2 :SIDE/3
PU RT 9Ø FD :SIDE/1Ø LT 9Ø PD
SETPC :COLORPANES
PANES: SIDE
PU BK :SIDE/3 PD
PANES :SIDE
PU LT 9Ø FD :SIDE/2 RT 9Ø PD
PANES :SIDE
PU FD :SIDE/3 + :SIDE/2 RT 9Ø FD :SIDE/1Ø + :SIDE/3 + :SIDE/2 RT 9Ø
  PD
HT
END
```

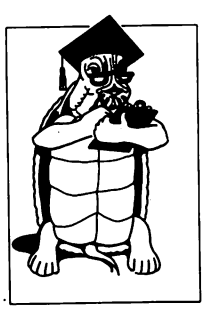
NOTE A space is mandatory before the use of the variable following a primitive command and as an input variable for a procedure. A space is unnecessary when the variable is read as part of an arithmetic operation. Any unnecessary spaces are included for ease of readability.

```
TO NEIGHBORHOOD :TIME
PU LT 9Ø FD 5Ø RT 9Ø PD
HOUSE2 3Ø 2 3 4 5
```

```
PRINT [ONE SMALL HOUSE IS ON THE STREET.]
PU RT 90 FD 50 LT 90 PD
HOUSE2 45 3 4 5 2
PR [SOMEONE BUILDS NEXT DOOR.]
PU RT 90 FD 65 LT 90 PD
HOUSE2 60 4 5 2 3
PR [NOW IT IS A NEIGHBORHOOD.]
PRINT []
PR [WATCH THE STRANGE COLOR EFFECTS!]
WAIT :TIME
FLASH2 :TIME
CLEARTEXT
PU HOME PD
END

NEIGHBORHOOD 100
```

NOTE Save these procedures in a file named COLORFUL.STUFF.

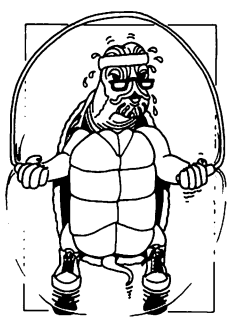
Figure 3-12.



Turtle Tester. Erase the current workspace and create a new file of procedures to draw a garage. Use variables for the size and color so the garage can be drawn in a variety of sizes and colors. Draw the garage on several background colors. Save the file under a name of your choice.

Turtle Teaser. Add the garages to the neighborhood drawn in TURTLE TRUTH.

Activity 3.4 Turtle Variables



Turtle Trials. This activity further explores the use of PenColor and BackGround color. It also introduces and distinguishes the terms "local variables" and "global variables."

1. Enter the following commands to load a previously created file from your file disk. Then add the following procedures to the files and observe the results. Describe your observations on the lines provided.

```
LOAD "PICTURE2
POTS

TO POT :COLORPOT
ST
SETPC :COLORPOT
RT 90 FD 15 RT 90
REPEAT 2 [FD 10 RT 90 FD 30 RT 90]
FD 10 RT 90 FD 5
REPEAT 4 [FD 20 LT 90]
BK 5 RT 90 FD 10 LT 90 FD 15 RT 90
HT
END

POT 1 _____
PX
```

```
PR PEN _____
POT 1 _____
POT 1 _____
PE
PR PEN _____
POT 1 _____
POT 1 _____
PD
PR PEN _____
ST
```

NOTE The command PEN outputs a two-part message describing the current state of the pen. The first part is either PENUP, PENDOWN, PENERASE, or PENREVERSE and the second part is the number of the pencolor.

2. Enter the following procedures and observe the different results when using PE on colored backgrounds. Describe the screen displays on the lines provided.

```
TO PE.DEMO :COLORPOT :COLORBG
SETBG :COLORBG
ERASEPOT :COLORPOT
END

TO ERASEPOT :COLORPOT
CLEARTEXT
POT :COLORPOT
PE
PR PEN
WAIT 1ØØ
POT :COLORPOT
PD
END

PE.DEMO 1 2 _____
PE.DEMO 2 3 _____
PE.DEMO 3 4 _____
PE.DEMO 4 5 _____
```

3. Use the editor to change the ERASEPOT and PE.DEMO to the procedures below. Notice you save considerable typing time by using the structure of one procedure, changing it slightly, and assigning it a new

name upon completion. After modifying the procedures, enter the commands and observe the results. Record your observations on the lines provided.

```
TO REVERSEPOT :COLORPOT
CLEARTEXT
POT :COLORPOT
PX
PR PEN
WAIT 1ØØ
POT :COLORPOT
WAIT 1ØØ
POT :COLORPOT
PD
END

TO PX.DEMO :COLORPOT :COLORBG
SETBG :COLORBG
REVERSEPOT :COLORPOT
END

PX.DEMO 1 2 _____
CS
PX.DEMO 2 3 _____
CS
PX.DEMO 3 4 _____
CS
PX.DEMO 4 5 _____
CS
SETBG Ø
SETPC 1
```

4. Enter the following procedure and notice the effect of PE and PX on a drawing that uses lines other than vertical and horizontal. Compare your PLANT with the one illustrated in Figure 3-13. Observe the screen display and record your observations on the lines provided.

```
TO PLANT :COLORPOT :COLORSTEM :COLORFLOWER
POT :COLORPOT
STEM :COLORSTEM
FLOWER :COLORFLOWER
END
```

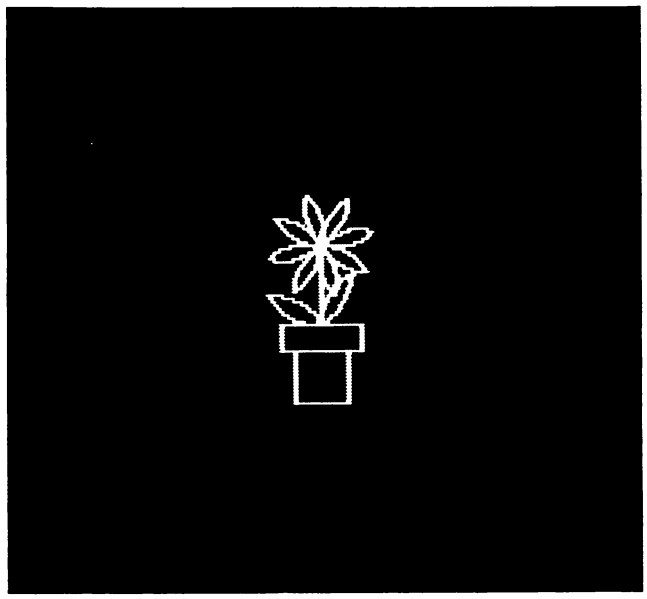


Figure 3-13.

```
PLANT 2 3 4 _____
PX
PLANT 2 3 4 _____
PX
PLANT 2 3 4 _____
PE
PLANT 2 3 4 _____
SETBG 5
CS
PLANT 2 1 0 _____
PX
PLANT 2 1 0 _____
PX
PLANT 2 1 0 _____
PE
PLANT 2 1 0 _____
PR PEN _____
SETBG Ø
ST
```

5. Enter the following procedures and observe the results. Compare your screen display to the illustration in Figure 3-14.

```
TO POSIE :COLORSTEM :COLORFLOWER
```

```
STEM :COLORSTEM
FLOWER :COLORFLOWER
END
```

POSIE 2 4 _____

```
TO VARIABLEDEMO
PR :COLORSTEM
PR :COLORFLOWER
END
```

VARIABLEDEMO _____

6. Enter the following procedure and describe the results.

```
TO ROW :COLORSTEM :COLORFLOWER
REPEAT 7 [POSIE :COLORSTEM :COLORFLOWER PU RT 90 FD 40 LT 90 PD]
END
```

ROW 2 5 _____

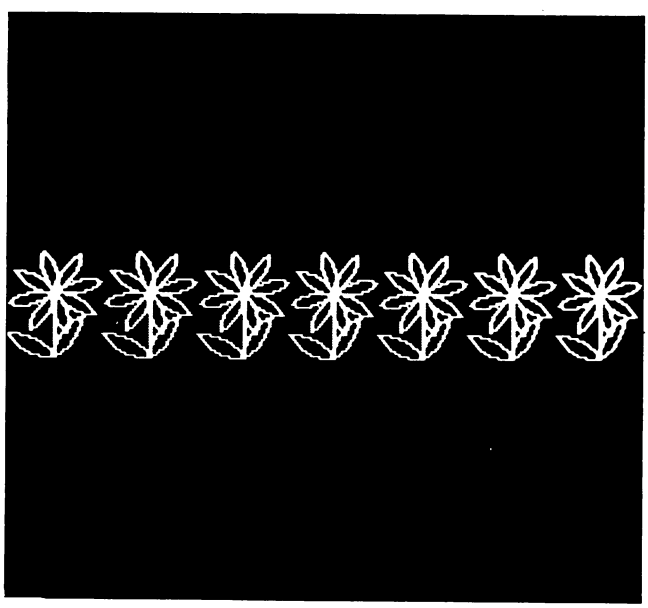


Figure 3-14.

7. Modify the procedure POSIE and describe the screen display on the lines provided.

```
TO POSIE
MAKE "COLORSTEM 2
MAKE "COLORFLOWER 4
STEM :COLORSTEM
```

```
FLOWER :COLORFLOWER
END

POSIE _____

VARIABLEDEMO _____
```

8. Enter the procedure ASSIGN and modify POSIE, STEM, FLOWER, and ROW as indicated below. Describe the screen display on the lines provided.

```
TO ASSIGN
MAKE "COLORSTEM 2
MAKE "COLORFLOWER 4
END

TO POSIE
STEM
FLOWER
END

TO STEM
SETPC :COLORSTEM
LT 90
LEAF
RT 90
LEAF
SETH 0
FD 30
END

TO FLOWER
SETPC :COLORFLOWER
REPEAT 8 [PETAL RT 360/8]
PU
BK 30
PD
END

TO ROW
ASSIGN
REPEAT 7 [POSIE PU RT 90 FD 40 LT 90 PD]
END

ROW _____
```

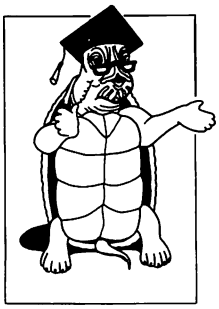
9. Modify ASSIGN and ROW as indicated below. Describe the screen display.

```
TO ASSIGN
MAKE "COLORSTEM 2
```

```
MAKE "COLORFLOWER RANDOM 6
END

TO ROW
REPEAT 7 [ASSIGN POSIE PU RT 9Ø FD 4Ø LT 9Ø PD]
END

ROW ____________________________________
```

10. When COLORFLOWER in Part 9 is assigned the color RANDOM 6, it can take on any value from 0 to 5 and display that color. It is possible that 0 could be selected and a black flower would be drawn on a black background. In order to avoid this possibility, make the following change in the ASSIGN procedure. Also, the following additions to the ROW procedure are useful in many instances to ensure that the pen color is returned to white and the pen is down after completing a diagram or picture. The TRACE procedure is included so you are aware of the turtle's state during the running of the procedure. Notice the use of a global variable to hold the value of the pen's original state. Type in the TRACE procedure, redefine POSIE as POSIE2, and edit ROW to read as given below. Describe the screen display.

```
TO ASSIGN
MAKE "COLORSTEM 2
MAKE "COLORFLOWER (1 + RANDOM 5)
END

TO TRACE
PR PEN
WAIT 15Ø
CLEARTEXT
END

TO POSIE2
STEM
TRACE
FLOWER
TRACE
END

TO ROW
MAKE "FIRSTPEN PEN
REPEAT 7 [ASSIGN POSIE2 PU TRACE RT 9Ø FD 4Ø LT 9Ø PD]
SETPEN :FIRSTPEN
TRACE
END

ROW ____________________________________
SAVE "COLORFUL.POSIES
ERPS
```



Turtle Talk. As a review, recall that PENDOWN (PD) and PENUP (PU) establish conditions for allowing the turtle to draw (PD) or not to draw (PU). There are two other commands which allow for the removal of a previously drawn diagram. PENERASE (abbreviated PE) retraces an existing diagram and erases it. PENREVERSE (abbreviated PX) combines the PENDOWN and PENERASE commands.

Part 1 demonstrates that PENREVERSE erases any previously drawn diagrams encountered in the turtle's path, but draws in places which are not previously drawn. Part 2 illustrates that these results occur only when using black and white. When color is introduced, strange results occur with the PENREVERSE command. The diagrams are not completely erased; instead, traces of dots and drawn lines remain. However, two successive PENREVERSE commands return the original drawing. While these peculiarities may be disconcerting, the Turtle Truth will illustrate an unusual display that makes advantageous use of them.

The PR PEN command displays the current status of the pen and the color that was most recently used.

In Logo, input variables consist of two types: local variables and global variables. All input variables that appear in Activities 3.1, 3.2, and 3.3 are classified as local variables.

Local variables are defined by prefixing a variable name with a colon (:) when defining a procedure. For example, in Part 5, TO POSIE :COLORSTEM :COLORFLOWER defines COLORSTEM and COLORFLOWER to be local variables.

An important characteristic of local variables is that they are defined only within the procedure which references them. Thus, COLORSTEM and COLORFLOWER are local variables for POSIE. VARIABLEDEMO demonstrates that COLORSTEM and COLORFLOWER are undefined, since they are not local variables for that procedure. Part 6 defines COLORSTEM and COLORFLOWER to be local variables for ROW, as well as POSIE.

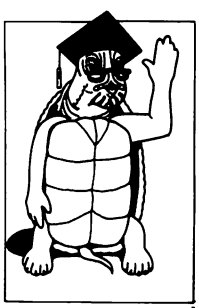
In contrast to local variables, global variables are defined by using the MAKE command. For example, in Part 7, MAKE "COLORSTEM 2 defines COLORSTEM to be a global variable which receives the value 2. Global variables differ from local variables by being defined outside of procedures which reference them, as well as in the procedure that defines the variable. VARIABLEDEMO demonstrates that COLORSTEM and COLORFLOWER have the values which are defined in POSIE, even though they are not explicitly defined in VARIABLEDEMO.

In Part 8, the ASSIGN procedure gives global variables COLORSTEM and COLORFLOWER the respective values 2 and 4. Since the revised version of ROW now includes ASSIGN, the COLORSTEM and COL-

ORFLOWER variables in POSIE are defined globally. Therefore, the inclusion of COLORSTEM and COLORFLOWER, when defining the respective STEM and FLOWER procedures, is unnecessary.

In Part 10, FIRSTPEN is globally assigned the value PEN, which is defined internally by the computer.

The RANDOM command in Part 9 permits an easy way to generate random numbers. RANDOM N generates whole numbers, inclusively, from 0 to N-1. For example, RANDOM 5 generates numbers of the form 0, 1, 2, 3, or 4. Since 0 is black, the random selection of a non-black color can be obtained by adding one to RANDOM 5 when it is called in the procedure, as occurs in Part 10. It generates numbers 1, 2, 3, 4, or 5.



Turtle Truth. Examine the following procedures and predict the resulting screen display. Enter the commands to verify your predictions and compare the final screen display to the picture illustrated in Figure 3-15.

```
TO PETALS :ARCLENGTH
MAKE "COLOR (1 + RANDOM 5)
SETPC :COLOR
REPEAT 2 [ARCR :ARCLENGTH 6Ø RT 12Ø]
CLEARTEXT
PR [THIS IS THE CURRENT STATE OF THE PEN]
PR PEN
WAIT 1ØØ
CLEARTEXT
END

TO ANYFLOWER :ARCLENGTH :MANY
REPEAT :MANY [PETALS :ARCLENGTH RT 36Ø/:MANY]
END

TO DAISY
MAKE "FIRSTPEN PEN
ANYFLOWER 5Ø 16
PE
ANYFLOWER 5Ø 8
PD
ANYFLOWER 35 8
ANYFLOWER 75 16
SETPEN. :FIRSTPEN
PR PEN
REPEAT 2Ø [SETBG RANDOM 6]
```

```
SETBG Ø
END

DAISY ____________________________________
SAVE "VIDEO.DAISY
```

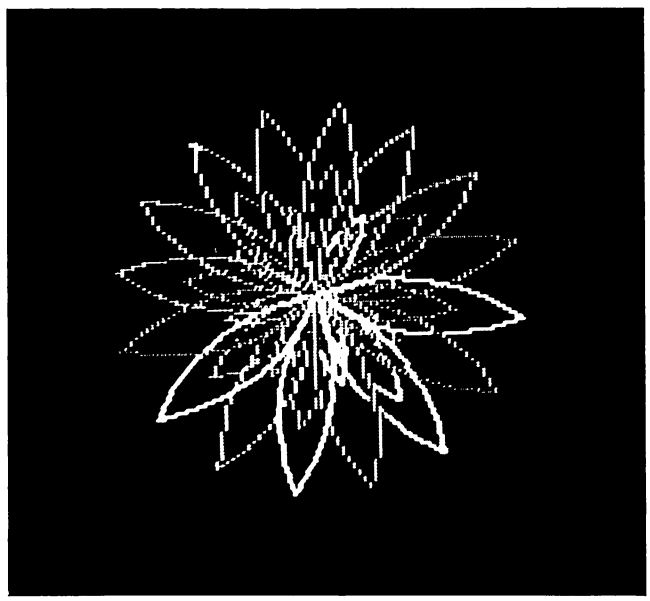
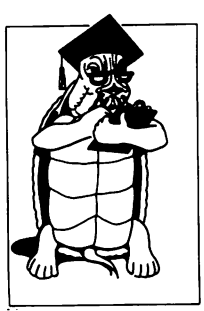
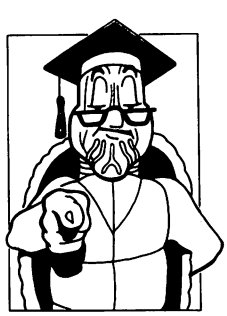


Figure 3-15.

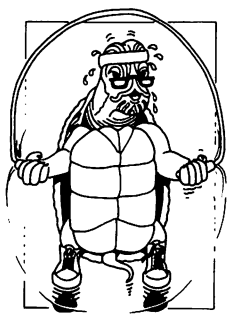


Turtle Tester. Modify the DAISY procedure so that it produces a new flower. Change the number of petals, the proportion of each layer to the others, and the printed message. Save it under a file name of your choice.



Turtle Teaser. Write a series of procedures to draw a row of flowers, with randomly colored petals, across the screen. Save these procedures under a file name of your choice.

Activity 3.5 Turtle Iterations



Turtle Trials. This activity introduces the concept of recursion.

1. Enter the following procedure and describe the result.

```
TO MESSAGEBOX
PR [TURTLE BOX]
END
MESSAGEBOX _____
```

2. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGE BOX
PR [TURTLE GRAPHICS]
PR [TURTLE GRAPHICS]
PR [TURTLE GRAPHICS]
END
MESSAGEBOX _____
```

3. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
CLEARTEXT
REPEAT 3 [PR [TURTLE GRAPHICS]]
END
MESSAGEBOX _____
```

4. Use the editor to modify MESSAGEBOX to read as follows and then describe the results on the lines provided.

```
TO MESSAGEBOX :NUMBER
CLEARTEXT
REPEAT :NUMBER [PR [TURTLE GRAPHICS]]
END
MESSAGEBOX 1Ø _____
MESSAGEBOX 1ØØØ
```

```
CTRL-G (after a while) _____________________
MESSAGEBOX 1000
CTRL-Z (after a while) _____________________
```

NOTE You will now receive the message

```
PAUSING . . .:
  JUST BEFORE LEAVING MESSAGEBOX
MESSAGEBOX ?■
```

At this point, type

```
CO
CTRL-W _____________________________________
```

NOTE At this point printing will stop, but no cursor will appear. Press any key to continue.

```
CTRL-G (after a while) _____________________
```

5. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
PR [TURTLE GRAPHICS]
MESSAGEBOX
END

MESSAGEBOX
CTRL-G _____________________________________
MESSAGEBOX
CTRL-Z _____________________________________
CO _________________________________________
CTRL-W _____________________________________
(Press any key)
CTRL-G _____________________________________
```

6. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
TYPE [TURTLE GRAPHICS]
MESSAGEBOX
END

MESSAGEBOX _________________________________
CTRL-G
```

7. Use the editor to modify MESSAGEBOX to read as follows:

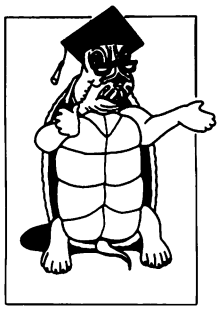
```
TO MESSAGEBOX
PR [TURTLE GRAPHICS]
REPEAT 3 [PR "]
MESSAGEBOX
END

MESSAGEBOX
CTRL-W _____
CTRL-Z _____
CTRL-G _____
CO _____
CTRL-G _____
```

8. Enter the following procedure and describe the results.

```
CLEARTEXT
TO DESIGN
ST
REPEAT 4 [FD 4Ø RT 9Ø]
PR [PROVERBIAL SQUARE]
END

DESIGN _____
```

9. Use the editor to modify DESIGN to read as follows, then describe the results.

```
TO DESIGN
ST
REPEAT 4 [FD 4Ø RT 9Ø]
PR [INFINITE SQUARE]
DESIGN
END

DESIGN
CTRL-G (after a while) _____
```



Turtle Talk. When you boot the computer with Logo, you are placed in full text mode. The placement of messages depends upon whether full text mode or mixed graphics and text mode is in effect. When in full text screen mode, all messages appear from the top of the screen to the bottom.

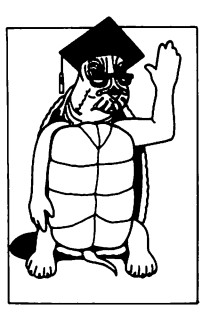
In contrast, the mixed graphics and text screen has a four-line text window located at the bottom of the screen. Parts 8 and 9 demonstrate that messages in this mode of operation appear first at the top of the text window, then move toward the bottom. Scrolling (when messages disappear from the text window) occurs when more than four lines of text are printed. The CLEARTEXT command removes all messages from the screen display, regardless of the screen display mode.

Parts 1 through 8 demonstrate a variety of methods by which the repetitive display of a message can appear on the full text screen. Part 5 introduces a technique known as recursion, which is the simplest method of accomplishing a repetitive display.

Recursion allows you to use the procedure name within the procedure so that it executes repeatedly. The difficulty which arises, however, is in stopping the execution of the procedure.

Part 5 also introduces three methods for stopping the execution of a procedure. By pressing CTRL-G, execution of the procedure is terminated and the user is returned to toplevel mode. CTRL-Z behaves as a pause, but when CO (COntinue) is typed, execution of the procedure continues. CTRL-W also stops program execution, but the program proceeds to execute when any key is typed. Activity 3.7 will teach you how to terminate the execution of a procedure that uses recursion by program control, rather than with the CTRL key functions.

Part 9 introduces the use of recursion in graphics mode. While the repetitive drawing of the same square in the same position may not appear to be particularly useful, Activity 3.6 will teach you how to generate some interesting graphic effects by using the principle of recursion.



Turtle Truth. Study the form and content of the following procedures. Type in the commands to verify your deductions. Compare the screen displays to Figures 3-16, 3-17, and 3-18.

```
TO AIR :COLOR
SETPC :COLOR
BASE
DECK
GUNS
TOWER
END

TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END

TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO BASE
ST
SETH 270
FD 40
SETH 135
FD 10
SETH 90
FD 60
SETH 45
FD 10
SETH 270
FD 40
END

TO DECK
BK 40
RECTANGLE 80 5
END

TO GUNS
SETH 0
FD 5
LT 90
FD 5
RECTANGLE 4 2
SETH 0
FD 2
RECTANGLE 6 2
END

TO TOWER
PU
SETH 270
FD 20
PD
RECTANGLE 16 8
```

```
SETH Ø FD 8
SETH 27Ø FD 2
RECTANGLE 12 6
SETH Ø FD 6
SETH 27Ø FD 2
RT 9Ø FD 6
BK 2 SETH 27Ø
PU BK 2 PD
FD 4 PU SETH 18Ø
FD 23 SETH Ø
PD HT
END

AIR 2 ____________________________________

TO CONVOY
SETBG 5
MAKE "COLOR RANDOM 2
REPEAT 3 [AIR :COLOR SETH Ø PU LT 9Ø FD 28Ø/3 PD]
END

CS
CONVOY ___________________________________
```

Edit CONVOY to read as follows:

```
TO CONVOY
SETBG 5
MAKE "COLOR RANDOM 2
REPEAT 3 [AIR :COLOR SETH Ø PU LT 9Ø FD 28Ø/3 PD]
PU SETH Ø FD 3Ø SETH 27Ø PD
CONVOY
END

CS
CONVOY
CTRL-L

CTRL-G (When you are tired of this)
SAVE "SHIPS
ERPS
```

NOTE The MAKE "COLOR command controls the color of each row of carriers. RANDOM 2 randomly selects either color 0 or 1. This ensures that the drawing is visible on the blue background.

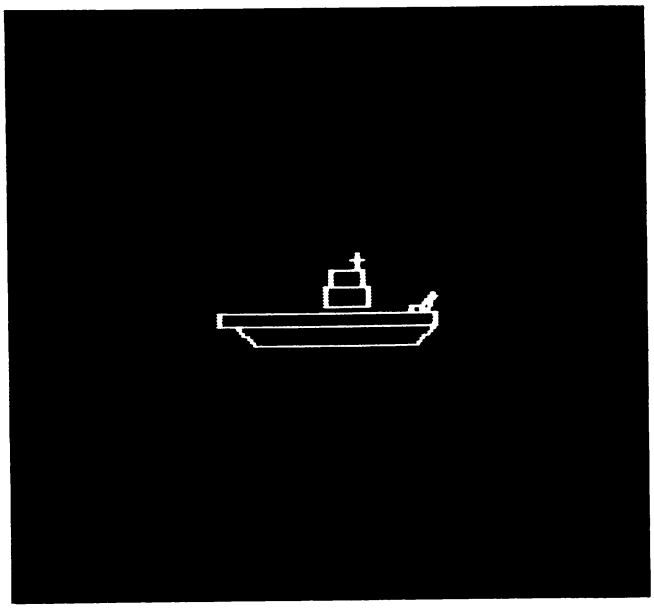


Figure 3-16.

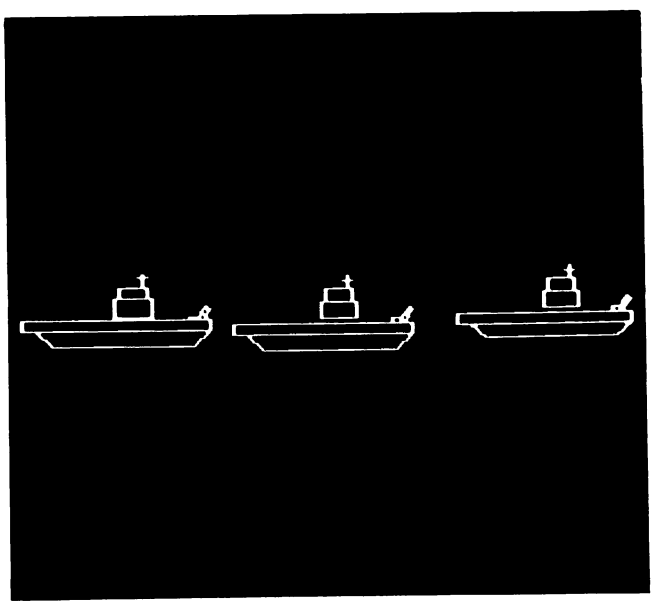


Figure 3-17.

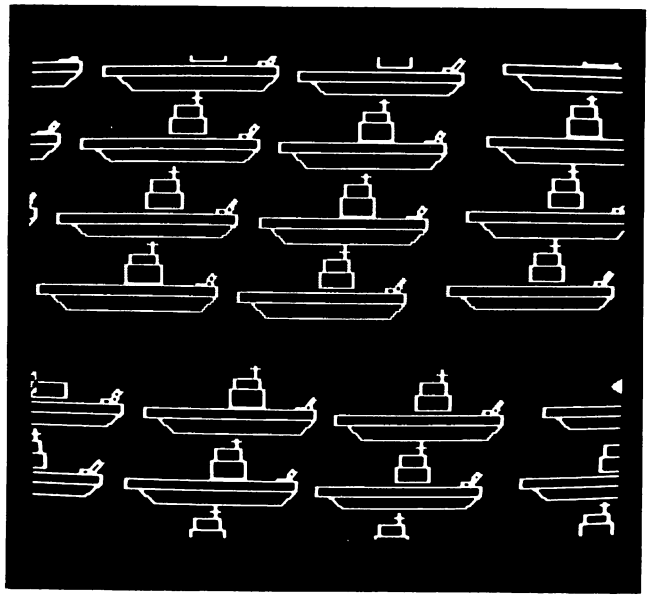
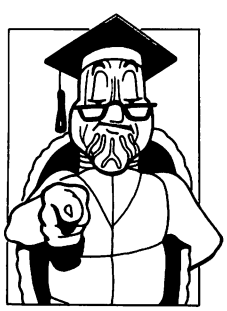


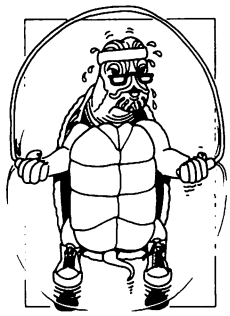
Figure 3-18.

Turtle Tester. Develop a series of procedures that recursively display an object of your choice across the screen. Include PenColor and BackGround color in conjunction with the MAKE command to generate an interesting display. Save the procedures under a suitable file name.



Turtle Teaser. Use the aircraft carrier or another object and devise a vertical recursive pattern.

Activity 3.6 Turtle Limits



Turtle Trials. This activity uses the recursion principle to draw designs that can be easily varied through the use of input variables.

1. Enter the following procedure and describe the results. (See if you can spot the intentional error.)

```
TO POLY :SIDE
FD :SIDE
RT 90
POLY
END
POLY 40 _____
```

2. Use the editor to modify POLY to read as follows.

```
TO POLY :SIDE
FD :SIDE
RT 90
POLY :SIDE
END
POLY 40
CTRL-G _____
CS
POLY 60
CTRL-G _____
```

3. Use the editor to modify POLY as indicated, then describe the results. Compare your screen displays to the illustrations in Figures 3-19, 3-20, and 3-21.

```
TO POLY :SIDE
FD :SIDE
RT 90
POLY :SIDE + 5
END

POLY 5
CTRL-G (after several seconds) _____
CS
FENCE
POLY 10
```

```
CS
WINDOW
POLY 15
CTRL-G (after several seconds) _____
CS
WRAP
POLY 12
CTRL-G (after several seconds) _____
```

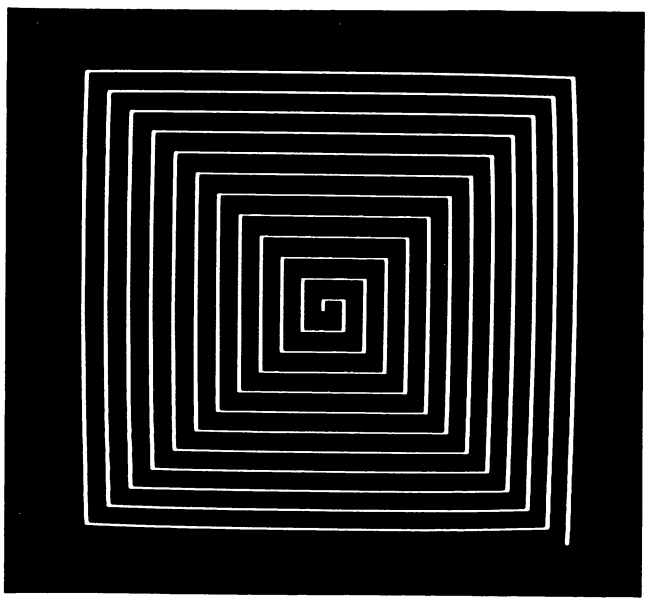


Figure 3-19.

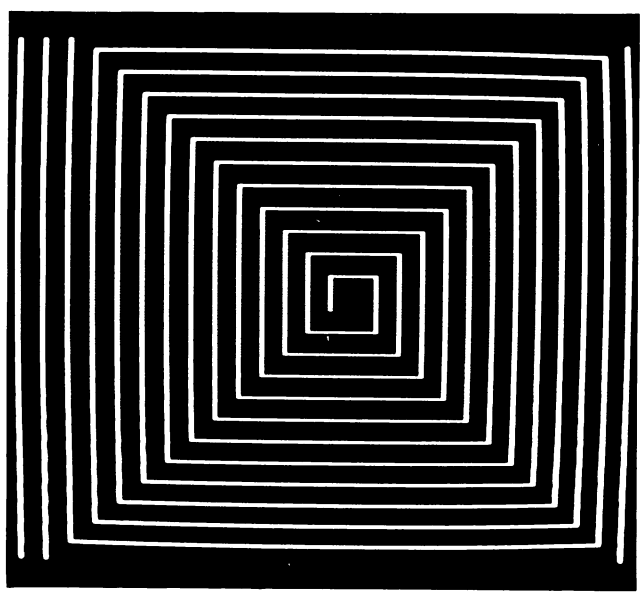


Figure 3-20.

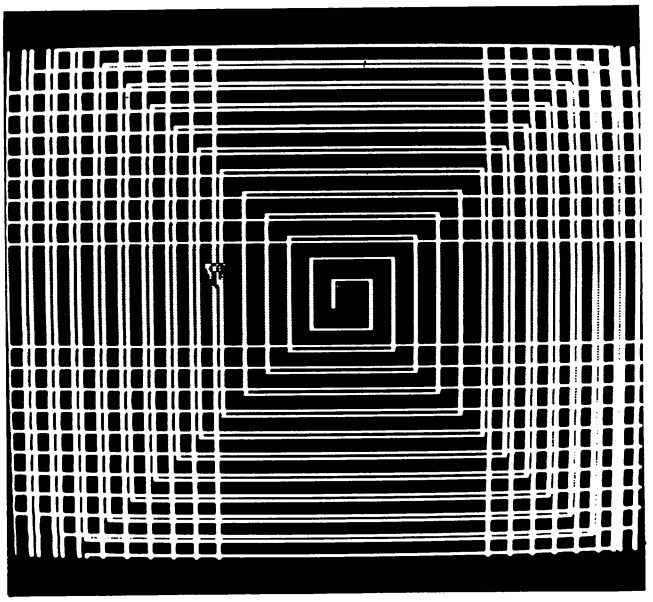


Figure 3-21.

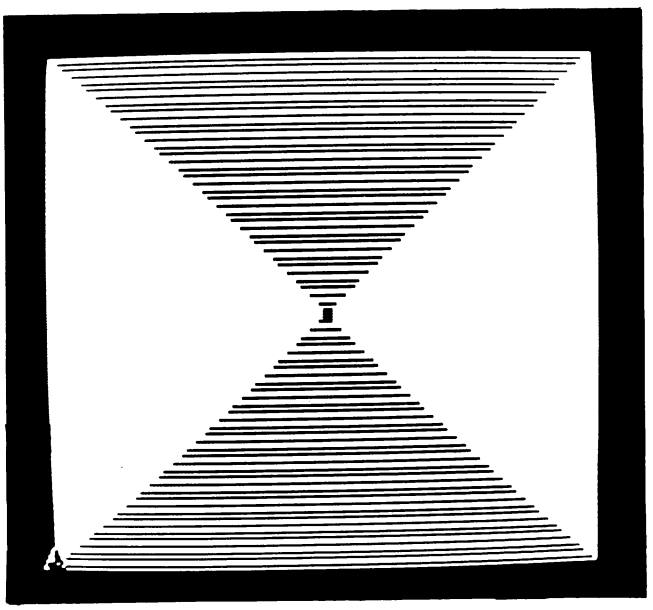


Figure 3-22.

4. Use the editor to modify POLY to read as follows. Then describe the results and compare your second screen display to the illustration in Figure 3-22.

```
TO POLY :SIDE :STEP
FD :SIDE
RT 90
POLY :SIDE + :STEP :STEP
END

FENCE
POLY 10 5 _____
CS
POLY 5 1 _____
```

5. Enter the new procedure HUE and insert it in POLY as shown below. Then type in the commands that call POLY and observe the interesting effects.

```
TO HUE
MAKE "COLOR (1 + RANDOM 5)
SETPC :COLOR
END

TO POLY :SIDE :STEP
HUE
FD :SIDE
RT 90
POLY :SIDE + :STEP :STEP
END

POLY 10 5 _____
CS
POLY 3 2 _____
CS
POLY 5 1 _____
CS
WRAP
POLY 10 3 _____
CTRL-G (when you tire of this)
```

6. Use the editor to modify POLY, observe the screen displays, and compare them to the diagrams in Figures 3-23 and 3-24. Describe the results on the lines provided.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
WAIT 10
POLY :SIDE + :STEP :STEP :ANGLE
END

FENCE
POLY 2 2 60 _____
CS
POLY 5 5 120 _____
```

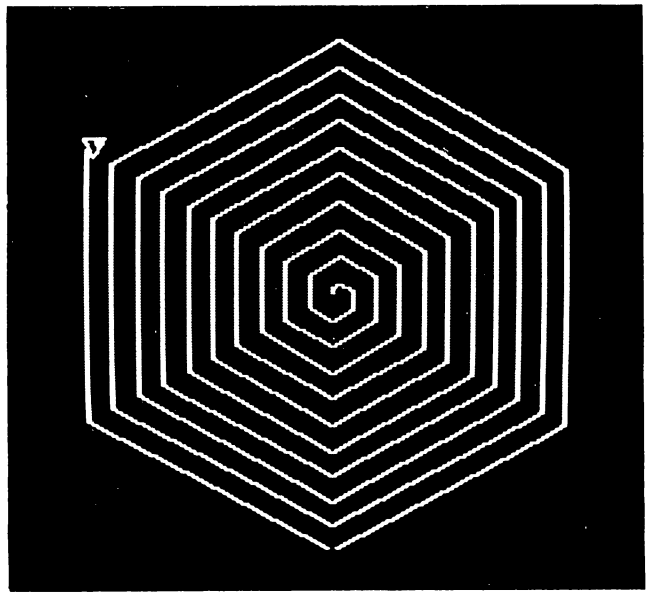


Figure 3-23.

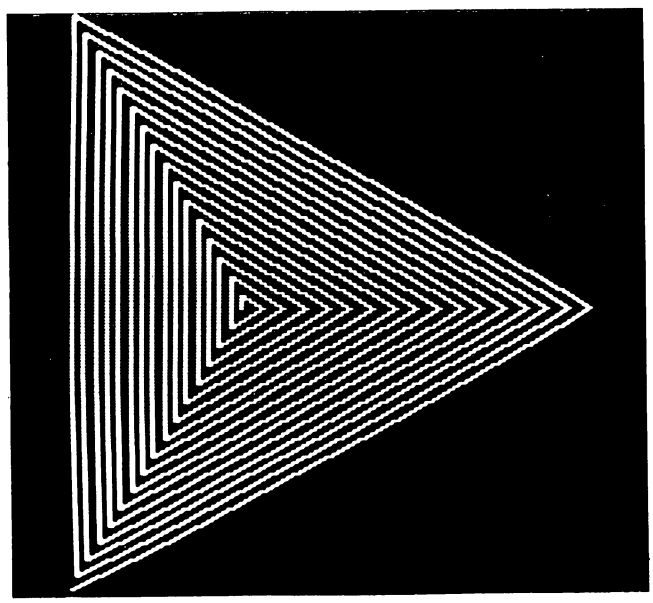


Figure 3-24.

7. Edit POLY so that it is modified to read as follows. Enter the TRACE procedure so that you can see the value of the angle change with each step. If the flashing message annoys you after a while, use CTRL-L to

view fullscreen graphics. Upon completion of each procedure, compare your screen displays to Figures 3-25 and 3-26.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
TRACE
POLY :SIDE :STEP :ANGLE + :STEP
END

TO TRACE
PR [THE PRESENT ANGLE VALUE IS]
PR :ANGLE
WAIT 30
CLEARTEXT
END

CS
POLY 20 10 120
CTRL-G (after a while) _____
CS
POLY 10 5 244
CTRL-G (after a long while) _____
```

8. Use the editor to modify POLY and TRACE to read as follows. Then enter the commands that call POLY and observe the results. The flashing message will be viewed after completion of the procedure. Use

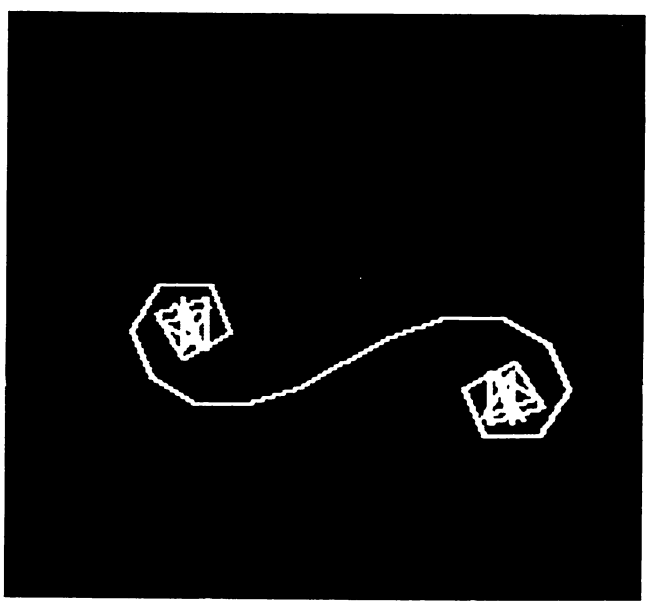


Figure 3-25.

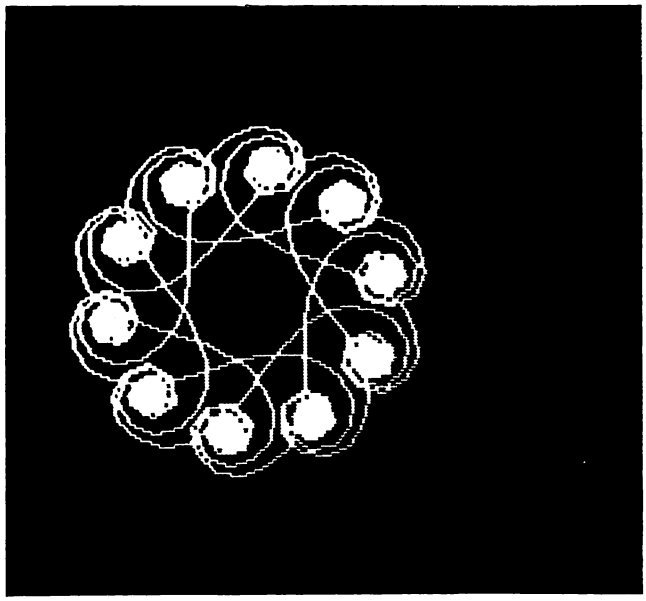


Figure 3-26.

CTRL-L if it bothers you. Compare your screen displays to the illustrations in Figures 3-27 and 3-28.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
TRACE
POLY :SIDE + :STEP :STEP :ANGLE + :STEP
END
TO TRACE
PR [CURRENT VALUES FOR EACH VARIABLE ARE]
PONS
WAIT 30
END
```

NOTE PONS stands for Print Out the NameS of all the variables and their values.

```
POLY 5 5 90 _____________________________________
CTRL-T _____________________________________
```

NOTE Notice the changes in the values of the variables each time POLY was executed. Observe that SIDE and ANGLE were incremented by five, which is the value of STEP each time. The value of COLOR changed randomly each time.

```
CTRL-S
CS
```

```
POLY 1 1 17 _____________________________________
CTRL-T _____________________________________
CS
WRAP
```

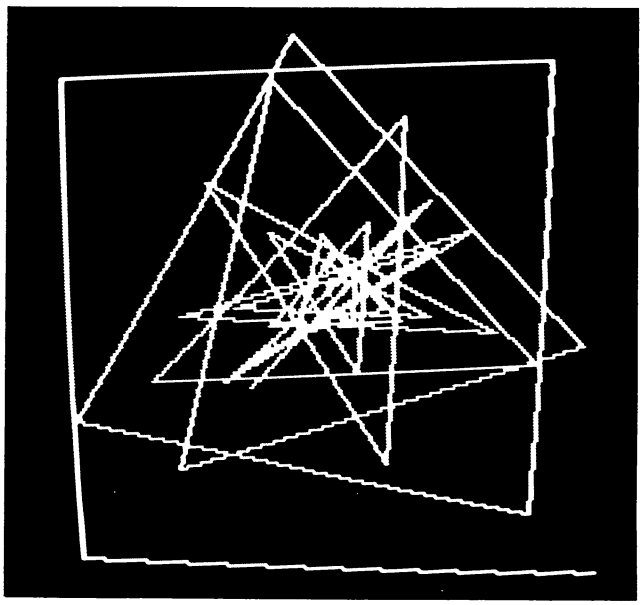


Figure 3-27.

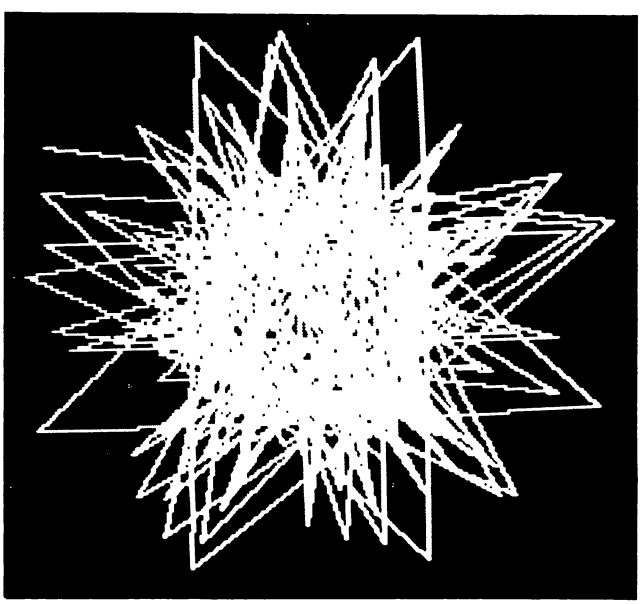
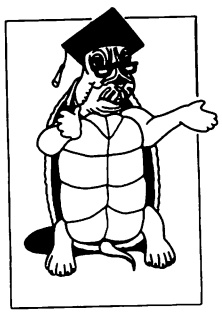


Figure 3-28.



Turtle Talk. Activity 3.5 introduced the concept of recursion with procedures that did not include input variables. This activity expands upon the last by defining recursive-type procedures which do include input variables.

Parts 1 and 2 demonstrate that when a procedure name with input variables is used recursively, the input variables must accompany the procedure name at all times, both when defining the procedure, and when using the procedure name within the procedure.

Parts 3 through 6 generate some interesting designs by changing the values of the input variables in the recursive component of the procedures. In Activity 3.5, a square appeared repetitively in the same position by recursively calling the procedure that draws a square. In contrast, Part 3 defines POLY :SIDE, where SIDE is increased by five before each successive execution of POLY, giving a spiral-like, box-shaped design, rather than the redrawing of the same object.

However, a new complication appears. The modifying of variables in recursive procedures creates a design that will eventually draw off the screen (as occurs in Part 3) and continue until the user terminates the execution of the program.

The CTRL-G function is a familiar standby for resolving this difficulty. Part 3 introduces some other alternatives. Preceding the execution of a procedure by the FENCE command causes Logo to give a TURTLE OUT OF BOUNDS error when the drawing occurs outside the screen boundaries. The WINDOW command allows the turtle to follow its path outside the screen boundaries without an error message, but only displays the drawing, which is within the screen dimensions.

WINDOW and FENCE commands remain in effect until memory is cleared or the WRAP command is used. The WRAP command puts Logo into its default mode, where drawings that appear outside the screen boundaries continue to draw (wrap around) in the indicated path. The WAIT command used in Part 6 slows down the speed at which the turtle draws a design.

Part 5 adds color to the procedure POLY. It is added by defining a procedure named HUE and including it as the first line in POLY. HUE randomly generates a PenColor to be used in POLY each time POLY executes. The color black (or 0) was eliminated through the use of (1 + RANDOM 5) in the procedure (the manner in which this was done is explained in Activity 3.4). Color not only adds interest, but allows the viewer to easily distinguish each side that is drawn.

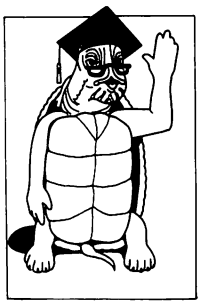
Parts 4 through 7 redefine POLY in terms of more than one input variable. A modification of these variables in recursively-defined procedures creates some fascinating and, perhaps, unexpected designs. It is interest-

ing to experiment with different input values to observe what other displays are possible.

In Part 7, a procedure named TRACE is added to POLY. It executes after each POLY and prints the value of ANGLE. This allows the viewer to watch ANGLE grow as it is incremented by the value of STEP.

Part 8 introduces a new workspace management command, PONS. PONS is included in the modification of TRACE so the values for every variable are printed after each execution of POLY. Because of the limited text window, you cannot read the text easily while the procedure is running. The use of CTRL-T after the procedure stops allows the viewer to study the display produced by PONS.

Activity 3.7 will show you how to control the graphics screen display with program control, rather than by using WINDOW, FENCE, or CTRL-G.



Turtle Truth. Study the procedures below and imagine the screen display. Type in the procedures and examine the effects of recursion and the use of incrementation. Upon completion, compare your screen displays to the illustrations in Figures 3-29 and 3-30.

```
TO OVAL :LENGTH :WIDTH
FD :LENGTH/2
ARCR :WIDTH/2 18Ø
FD :LENGTH
ARCR :WIDTH/2 18Ø
FD :LENGTH/2
END

CS
OVAL 5Ø 1Ø
CS

TO OVALS :LENGTH :WIDTH
REPEAT 6 [OVAL :LENGTH :WIDTH RT 6Ø]
END

OVALS 5Ø 1Ø
CS
TO MESSAGE
PR [THE OVAL'S LENGTH AND WIDTH ARE]
PR :LENGTH
PR :WIDTH
WAIT 1ØØ
```

```
CLEARTEXT
END

TO DESIGN :LENGTH :WIDTH :STEP
HUE
OVALS :LENGTH :WIDTH
MESSAGE
DESIGN :LENGTH + :STEP :WIDTH + :STEP :STEP
END

FENCE
DESIGN 50 10 40
CS
DESIGN 80 30 20
CS
```

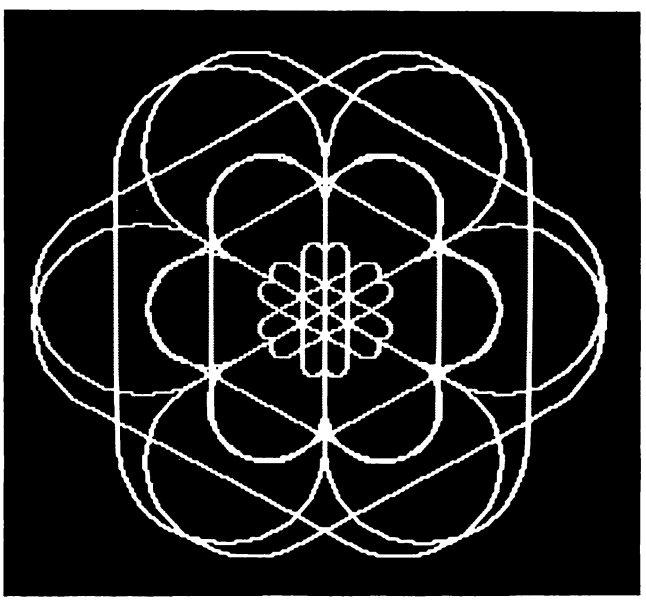


Figure 3-29.

NOTE Save these procedures in a file named RECURSION.PICS. Add the following Turtle Tester and Turtle Teaser procedures to the file RECURSION.PICS or save them separately under another name.

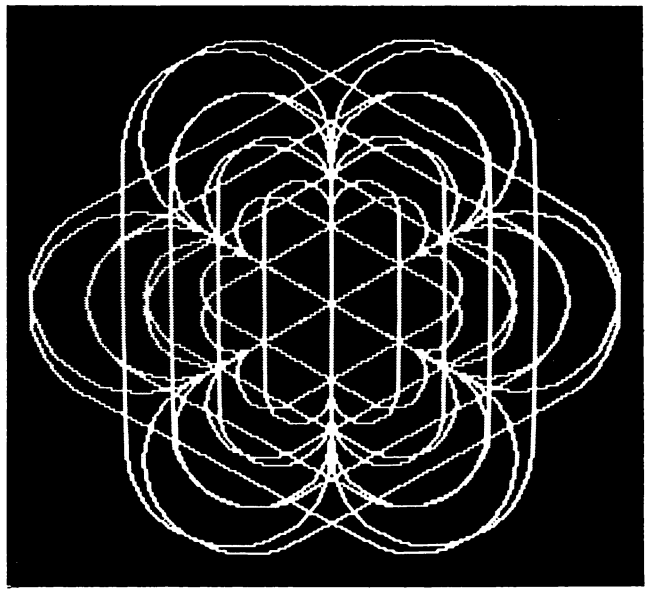


Figure 3-30.

Turtle Tester. Use the CIRCLES procedures listed below to create the illustration in Figure 3-31.

```
TO CIRCLES :RADIUS
REPEAT 5 [CIRCLER :RADIUS RT 72]
END
```

Create a recursive procedure named MANY.CIRCLES that draws CIRCLES with an initial radius of 30 with an increment of 10. Use FENCE in MANY.CIRCLES to stop the procedure. Figure 3-32 shows the partial completion of MANY.CIRCLES. Return the screen to WRAP mode when finished. (Answer on page 232.)

Turtle Teaser. Add color and a message to the MANY.CIRCLES procedure. The message should print the current radius of the circle and include a WAIT command permitting the user time to read the output.

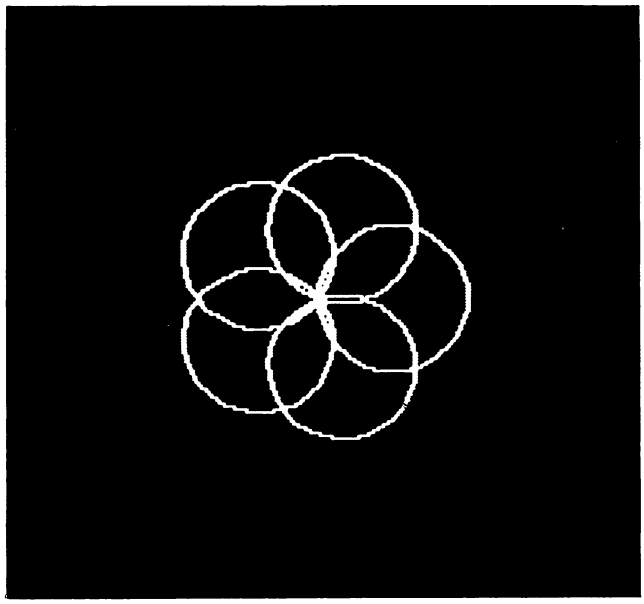


Figure 3-31.

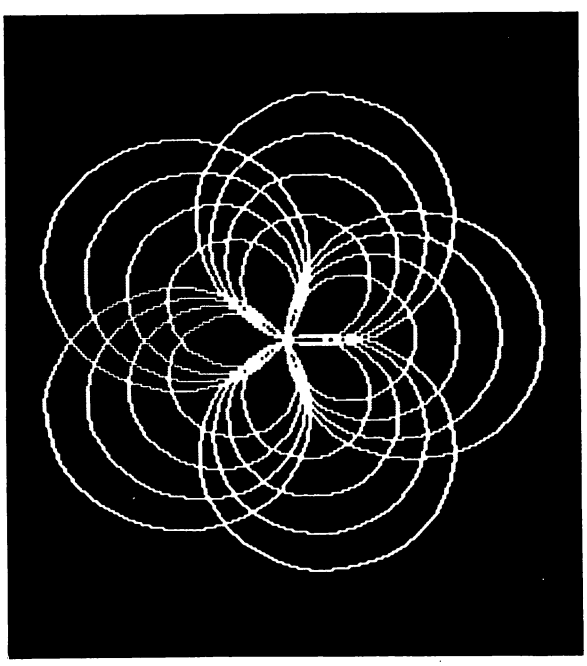
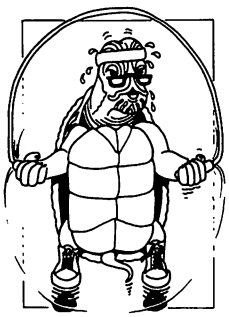


Figure 3-32.

Activity 3.7 Turtle Conditions



Turtle Trials. This activity shows you how to gain control of procedures that include simple uses of recursion.

1. Enter the following procedure and describe the result.

```
CS
TO TALLY :NUMBER
TEXTSCREEN
PR :NUMBER
TALLY :NUMBER + 1
END

TALLY 1
CTRL-G (after 10 seconds) _____
```

2. Modify TALLY to read as follows and describe the result.

```
TO TALLY :NUMBER
TEXTSCREEN
IF :NUMBER = 21 [STOP]
PR :NUMBER
TALLY :NUMBER + 1
END

TALLY 1 _____
```

NOTE The symbols = (equal), > (greater than), and < (less than) are used in predicates. Predicates are expressions which return the value TRUE or FALSE. For example, 6 > 3 is a predicate with the value TRUE and 7 < 4 is a predicate with the value FALSE.

3. Enter the following commands and record your observations below.

```
CLEARTEXT
TO MANYBOX :SIDE
REPEAT 4 [FD :SIDE RT 90]
MANYBOX :SIDE + 2
END

MANYBOX 10
CTRL-G (after 15 seconds) _____
```

4. Modify MANYBOX to read as indicated below and describe the results. Compare your second screen display to the illustration in Figure 3-33.

```
TO MANYBOX :SIDE
IF :SIDE > 1ØØ [STOP]
REPEAT 4 [FD :SIDE RT 9Ø]
MANYBOX :SIDE + 2
END

MANYBOX 1Ø _____________________________________
CS
MANYBOX -1ØØ ___________________________________
CTRL-L
HT
CS
CTRL-T
ST
```

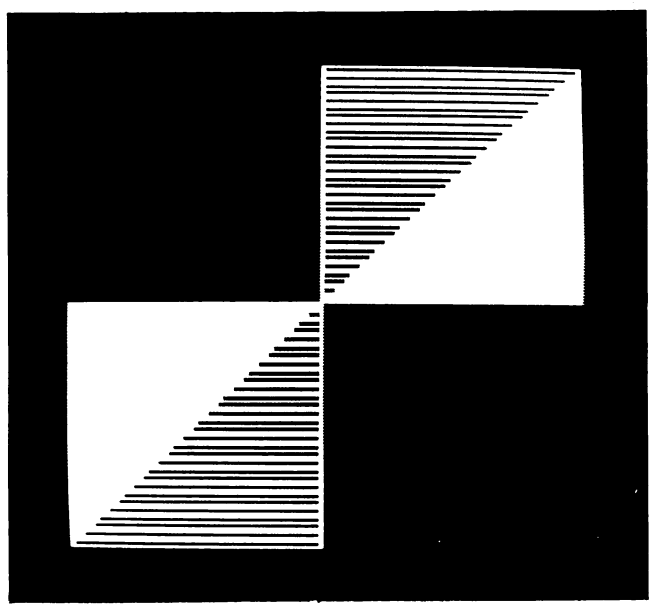


Figure 3-33.

5. Use the editor to modify the MANYBOX procedure to add color and a PRINT statement. The new procedure should read as follows:

```
TO MANYBOX2 :SIDE :COLOR
IF :COLOR > 5 [MAKE "COLOR Ø]
SETPC :COLOR
IF :SIDE > 7Ø [STOP]
REPEAT 4 [FD :SIDE RT 9Ø]
PR [PENCOLOR AND SIDE LENGTH ARE NOW]
PR :COLOR
```

```
PR :SIDE
WAIT 75
MANYBOX2 :SIDE + 5 :COLOR + 1
END

MANYBOX2 -7Ø Ø _____________________________
```

NOTE COLOR becomes a local variable when MANYBOX2 is defined. The IF :COLOR > 5 [MAKE "COLOR 0] command in MANYBOX2 does not cause COLOR to become a global variable. As an exception to the definition given in Activity 3.4, MAKE defines a global variable only if it has not been previously defined as a local variable.

6. Enter the following procedures and record the results.

```
TO POLY :SIDE :STEP :ANGLE
SETBG 1
IF :SIDE > 1ØØ [STOP]
MAKECOLOR
SETPC :COLOR
FD :SIDE
RT :ANGLE
POLY :SIDE + :STEP :STEP :ANGLE
END

TO MAKECOLOR
MAKE "COLOR RANDOM 6
IF :COLOR = 1 [MAKECOLOR]
END

POLY 2 4 72 _____________________________
CS
POLY 1Ø 5 12Ø _____________________________
CS
POLY 5 4 245 _____________________________
CS
POLY 8 1 315 _____________________________
SETBG Ø
```

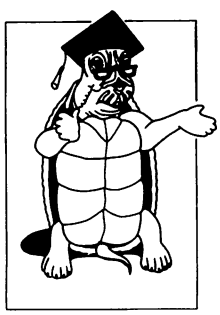
7. Modify POLY to read as follows and describe the results.

```
TO POLY :SIDE :STEP :ANGLE :NUMBER
IF :NUMBER = Ø [STOP]
FD :SIDE
RT :ANGLE
POLY :SIDE :STEP :ANGLE + :STEP :NUMBER - 1
END

POLY 5 5 244 1Ø _____________________________
CS
POLY 5 5 244 2ØØ _____________________________
```

```
CS
POLY 5 5 244 1000 _____________________________
```

8. Edit POLY to read as follows and describe the resulting screen displays.

```
TO POLY :SIDE :STEP :ANGLE :NUMBER
IF :NUMBER < 0 [STOP]
FD :SIDE
RT :ANGLE
PR :NUMBER
POLY :SIDE :STEP :ANGLE + :STEP :NUMBER - 1
END

POLY 5 5 244 1000 _____________________________
CS
POLY 5 5 244 670 _____________________________
CS
POLY 4 4 180 120 _____________________________
CS

PU RT 90 FD 138 LT 90 PD
POLY 4 4 330 620 _____________________________
```



Turtle Talk. The IF statement is a conditional expression. It provides a test to determine whether a condition is true or false. If the condition tested is true, the program execution is passed to the list enclosed in the brackets following the expression. If the tested condition is false, the program execution passes to the next line in the procedure.

When using recursion, the IF statement provides a means of programming the termination of a procedure. Part 1 presents a recursive procedure which displays the counting numbers beginning with one and will continue forever, if permitted. Part 2 adds the IF command to TALLY, achieving program control.

TALLY 1 begins execution of the program and displays the number 1 first. The statement TALLY :NUMBER + 1, within the procedure TALLY :NUMBER, recursively calls upon TALLY :NUMBER, but with an input value of one more than the previous call. This recursion has the effect of performing successively TALLY 1, TALLY 2, TALLY 3,...TALLY 20, if the procedure TALLY is written as follows:

```
TO TALLY :NUMBER
PR :NUMBER
```

```
TALLY :NUMBER + 1
END
```

By using recursion, the desired outcome is performed more simply and efficiently. In TALLY, the value assigned to NUMBER increases by 1 with each repetition. When NUMBER reaches a value of 21, the program terminates because of the STOP command within the brackets of the IF statement. Until then, the condition in the IF statement is false; thus the flow of the program proceeds to the next statement, PR :NUMBER, where the value of NUMBER is displayed.

The STOP command, because of its denotation, does not have the effect on the program you might expect. It does not "stop" the program. Instead, it returns control to wherever the procedure originally is called. Since TALLY is entered from toplevel mode, the STOP command returns the program to toplevel, giving the appearance of stopping the program. But if TALLY has been called upon by another procedure, program control would be returned to that procedure. Activity 3.8 expands your understanding of this concept when more complex situations using recursion are presented.

Parts 4 through 8 demonstrate the use of the IF statement to control recursive procedures which involve graphics. Three types of conditions that can be tested are used in predicates. Part 2 demonstrates = (equal), which tests whether two numbers are equal. Part 4 uses > (greater than), which tests whether the number on the left is greater than the number on the right. Similarly, < (less than) in Part 8 is demonstrated.

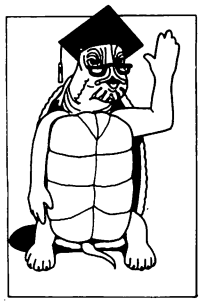
Part 6 also introduces a procedure which assures that the PenColor is not identical to the background color. MAKECOLOR uses recursion to randomly select a color number from 0 through 6, inclusively. If 1 (the background color) is selected, MAKECOLOR calls upon itself to generate another number until 1 is not obtained. When a number other than 1 is generated, the POLY procedure continues. Notice MAKECOLOR defines COLOR to be a global variable and, thus, it need not be defined in POLY.

In Parts 7 and 8, one becomes curious as to what designs can be created by experimenting or "playing" with input values, by modifying variables used in the graphic recursion procedures.

In Part 7, POLY 5 5 244 10 is not particularly exciting. POLY 5 5 244 200 generates a design of more interest, but it is not complete. POLY 5 5 244 1000 not only completes the design, but it also begins to redraw it.

The PR :NUMBER command, inserted in the editing of POLY in Part 8, provides a means by which the proper value for NUMBER in POLY can be determined. POLY 5 5 244 1000 now displays a value of approximately 330 when the diagram is completed, suggesting that 670 (1000 − 330) is an appropriate value for NUMBER to obtain the completed diagram.

Other input values for POLY in Part 8 generate designs which might be of interest to the reader.



Turtle Truth. Often the creation of a Logo procedure is actually a multi-step process. The original idea is accomplished through the building of an initial procedure, but upon reflection, a certain series of commands appear in several places. This pattern suggests that an additional procedure could replace these commands and produce a more concise product. As one studies the manner in which variables are defined, it may suggest that the use of global variables is more effective than the use of local variables. Study the following procedures as they evolve into a final form. Identify where procedures are obtaining values for variables. Compare your screen displays to the illustrations in the figures below each section.

```
TO TRI :SIDE :STEP
IF :SIDE < 3 [STOP]
REPEAT 3 [FD :SIDE LT 12Ø]
TRI :SIDE - :STEP :STEP
END

TRI 1ØØ 3 _____
CS
TRI 1ØØ 5 _____
CS

TO START :SIDE :STEP
SETPC 5
RT 9Ø
TRI :SIDE :STEP
END

START 12Ø 5 _____
CS

TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP [STOP]
REPEAT :MANY [FD :SIDE LT 36Ø/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

FILL 1ØØ 3 3 _____
CS
FILL 1ØØ 1 3 _____
CS
FILL 6Ø .5 6 _____
```

NOTE The FILL procedure shades or "fills" polygons with different degrees of density, depending upon the value for STEP. Figures 3-34, 3-35, and 3-36 illustrate some possible effects.

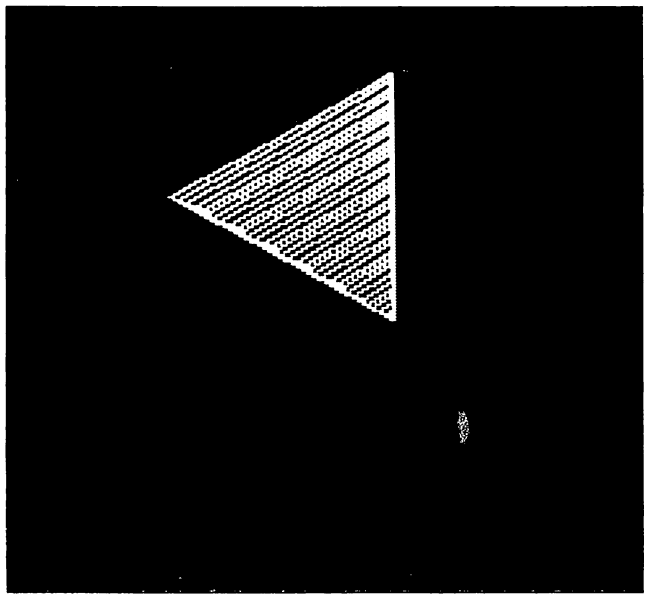


Figure 3-34.

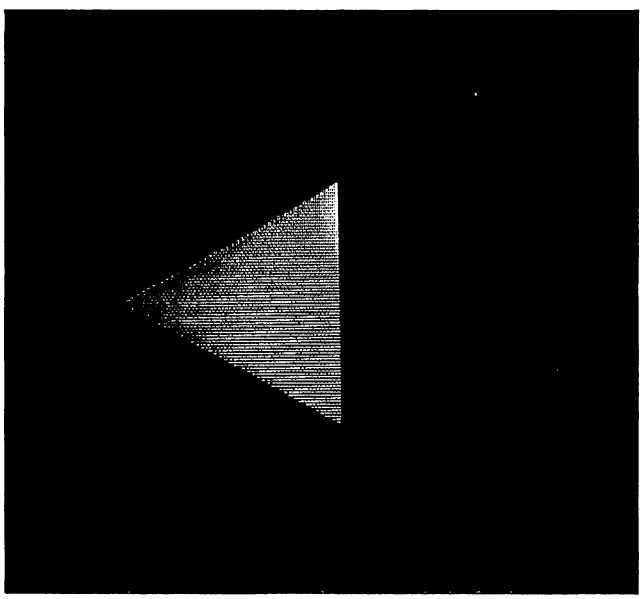


Figure 3-35.

```
TO COLORFILL :SIDE :STEP :MANY :COLOR
SETPC :COLOR
FILL :SIDE :STEP :MANY
END
```

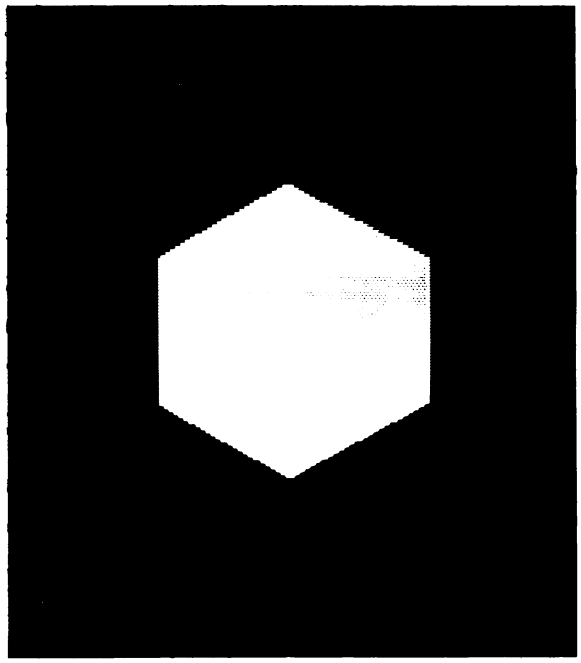


Figure 3-36.

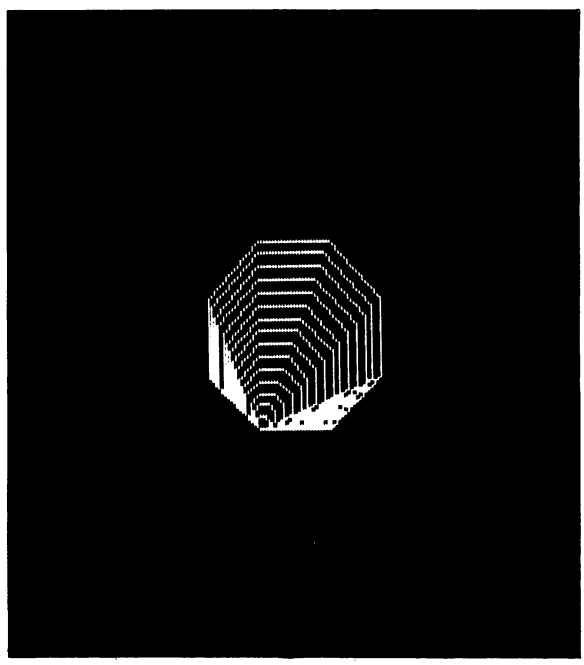


Figure 3-37.

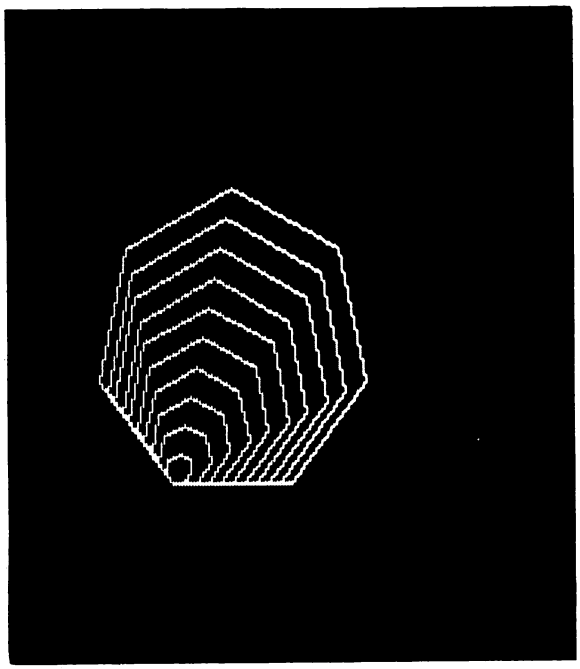


Figure 3-38.

```
COLORFILL 3Ø 2 8 4 _____________________________
CS
COLORFILL 5Ø 5 7 2 _____________________________
CS


TO HOUSE :SIDE :COLORHOUSE :COLORROOF
SETPC :COLORHOUSE
MAKE "MANY 4
MAKE "COLOR :COLORHOUSE
MAKE "STEP 1
RT 9Ø
COLORFILL :SIDE :STEP :MANY :COLOR
LT 9Ø
FD :SIDE
SETPC :COLORROOF
MAKE "MANY 3
MAKE "COLOR :COLORROOF
MAKE "STEP 3
RT 9Ø
COLORFILL :SIDE :STEP :MANY :COLOR
PU LT 9Ø BK :SIDE PD
END

HOUSE 5Ø 5 2

LT 9Ø PU FD 85 RT 9Ø PD
```

```
HOUSE 65 3 4

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH Ø PD
END

MOVE 9Ø 155

HOUSE 25 4 1

MOVE 27Ø 21Ø

HOUSE 35 2 5

TO HOUSES
HOUSE 5Ø 5 2
MOVE 27Ø 85
HOUSE 65 3 4
MOVE 9Ø 155
HOUSE 25 4 1
MOVE 27Ø 2Ø8
HOUSE 35 2 5
PU HOME PD
HT
END

CS
HOUSES
```

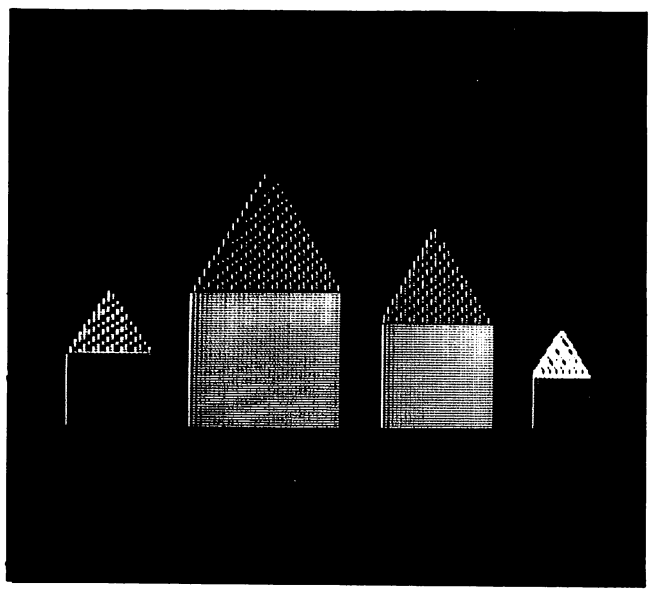


Figure 3-39.

NOTE The MAKE command in procedure HOUSE declares MANY, COLOR, and STEP to be global variables. Therefore, these input variables need not be included in the procedure headings for FILL and COLORFILL. Edit FILL, COLORFILL, and HOUSE to read as indicated below. Enter HOUSES to determine if the same screen display appears.

```
TO FILL :SIDE
IF :SIDE <:STEP [STOP]
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP
END

TO COLORFILL :SIDE
SETPC :COLOR
FILL :SIDE
END

TO HOUSE :SIDE COLORHOUSE :COLORROOF
SETPC :COLORHOUSE
MAKE "MANY 4
MAKE "COLOR :COLORHOUSE
MAKE "STEP 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
SETPC :COLORROOF
MAKE :MANY 3
MAKE "COLOR :COLORROOF
MAKE "STEP 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END

CS

HOUSES
```

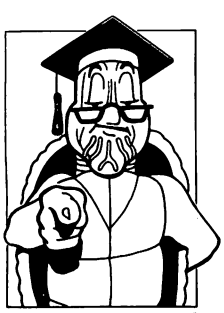
NOTE You might have observed that there is one more improvement to be made in HOUSE. MANY, COLOR, and STEP are defined twice in HOUSE. A better Logo structure is to define a procedure to perform this task. Modify HOUSE and enter GLOBALVALUES to read as follows. Enter HOUSES to verify that the same screen display occurs.

```
TO GLOBALVALUES :FIRST :SECOND :THIRD
MAKE "MANY :FIRST
MAKE "COLOR :SECOND
```

```
MAKE "STEP :THIRD
END

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
SETPC :COLORHOUSE
GLOBALVALUES 4 :COLORHOUSE 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
SETPC :COLORROOF
GLOBALVALUES 3 :COLORROOF 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END

HOUSES
```

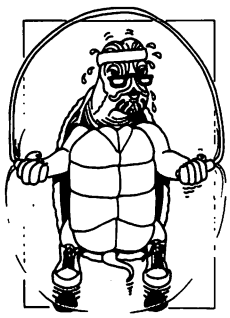
Save these procedures in a file named SOLID.PICS. Add the following Turtle Tester and Turtle Teaser procedures to the file SOLID.PICS or save them separately under another name.

Turtle Tester. Use the COLORFILL and MOVE procedures to position solid-colored shapes in the four corners of the screen. Put a purple pentagon in the upper left corner, an orange octagon in the upper right corner, a blue hexagon in the lower right corner, and a green circle in the lower left corner. (Answer on page 232.)



Turtle Teaser. Draw a robot with each section in a different color. Incorporate the COLORFILL procedure.

Activity 3.8 Turtle Tricks



Turtle Trials. This activity demonstrates how to develop procedures that include more complex uses of recursion.

1. Enter the procedure and describe the results.

```
TO TALLY :NUMBER
IF :NUMBER = 1Ø [STOP]
PR :NUMBER
TALLY :NUMBER + 1
END

TALLY 1 _____
```

2. Modify TALLY to read as follows and record the result.

```
TO TALLY :NUMBER
IF :NUMBER = 1Ø [STOP]
TALLY :NUMBER + 1
PR :NUMBER
END

TALLY 1 _____
```

3. Type the following procedures as indicated and describe the results.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 9Ø]
END

TO CHANGESQUARE :SIDE
IF :SIDE < 4 [STOP]
SQUARE :SIDE
CHANGESQUARE :SIDE*.5
END

CHANGESQUARE 12Ø _____
```

4. Modify CHANGESQUARE to read as follows and describe the result.

```
TO CHANGESQUARE :SIDE
IF :SIDE < 4 [STOP]
```

```
CHANGESQUARE :SIDE*.5
SQUARE :SIDE
END

CHANGESQUARE 12Ø _____
CS
```

5. Enter the following procedure and describe the result.

```
TO GYROSQUARE :SIDE
IF :SIDE < 8 [STOP]
SQUARE :SIDE
FD :SIDE/2 RT 45
GYROSQUARE :SIDE*.7Ø7
HT
END

GYROSQUARE 12Ø _____
```

6. Modify GYROSQUARE to read as follows and record the result.

```
TO GYROSQUARE :SIDE
IF :SIDE < 8 [STOP]
GYROSQUARE :SIDE*.7Ø7
SQUARE :SIDE
LT 45 BK :SIDE*.7Ø7
HT
END

GYROSQUARE 12Ø _____
```

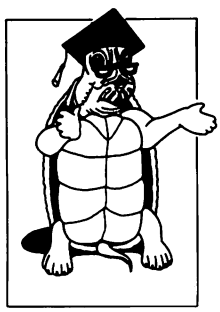
7. Edit GYROSQUARE to read as follows and record your observations.

```
TO GYROSQUARE :SIDE :NUMBER
IF :SIDE < 8 [STOP]
GYROSQUARE :SIDE*.7Ø7 :NUMBER
SQUARE :SIDE
IF :SIDE = :NUMBER [STOP]
LT 45 BK :SIDE*.7Ø7
HT
END

GYROSQUARE 12Ø _____
GYROSQUARE 12Ø 12Ø _____
```

8. Edit GYROSQUARE to add pencolor and a delay to the procedure. The change in pencolor before each execution of the procedure assists you in observing the sequence of commands that produce each square.

```
TO GYROSQUARE :SIDE :NUMBER
IF :SIDE < 8 [STOP]
GYROSQUARE :SIDE*.7Ø7 :NUMBER
MAKE "COLOR (1 + RANDOM 5)
```

```
SETPC :COLOR
SQUARE :SIDE
WAIT 1ØØ
IF :SIDE = :NUMBER [STOP]
LT 45 BK :SIDE*.7Ø7
END

GYROSQUARE 15Ø 15Ø
CS
GYROSQUARE 175 175
CTRL-L
CS
```



Turtle Talk. In Activities 3.6 and 3.7, the concept of simple recursion was introduced. Part 1 of this activity provides another example of simple recursion, while the example in Part 2 introduces complex recursion. Table 3-2 illustrates the output generated by each of these parts.

Table 3-2. Output generated by Parts 1 and 2.

Part 1 (Simple Recursion)	*Part 2 (Complex Recursion)*
TALLY 1 ----- 1	TALLY 1 ----- TALLY 2 (no output)
TALLY 2 ----- 2	TALLY 2 ----- TALLY 3 (no output)
TALLY 3 ----- 3	TALLY 3 ----- TALLY 4 (no output)
TALLY 4 ----- 4	TALLY 4 ----- TALLY 5 (no output)
TALLY 5 ----- 5	TALLY 5 ----- TALLY 6 (no output)
TALLY 6 ----- 6	TALLY 6 ----- TALLY 7 (no output)
TALLY 7 ----- 7	TALLY 7 ----- TALLY 8 (no output)
TALLY 8 ----- 8	TALLY 8 ----- TALLY 9 (no output)
TALLY 9 ----- 9	TALLY 9 ----- TALLY 1Ø(calls TALLY 9 because of STOP)
TALLY 1Ø ---- ?	TALLY 9 ----- 9
(program terminates)	TALLY 8 ----- 8
	TALLY 7 ----- 7
	TALLY 6 ----- 6
	TALLY 5 ----- 5
	TALLY 4 ----- 4
	TALLY 3 ----- 3
	TALLY 2 ----- 2
	TALLY 1 ----- 1
	?
	(program terminates)

With simple recursion, as in Part 1, each recursive call is preceded by the printing of value NUMBER in TALLY. For example, TALLY 1 first prints a "1" before calling TALLY 2, which prints a "2." Similarly, the numbers three through nine are printed.

When the STOP command is executed (for TALLY 10), program execution is transferred to the command following the last recursion, namely the END command, which terminates the program.

In contrast to Part 1, since the PR command in Part 2 follows the recursive formula TALLY :NUMBER + 1, the various recursion calls accumulate and define separate libraries or memory locations associated with the different values for NUMBER. Notice no printing occurs until after the STOP command executes for TALLY 10. At this point, program execution is transferred to the most previously-called recursion (TALLY 9) and proceeds to execute the commands (in this case, PR :NUMBER) following TALLY :NUMBER + 1. This process continues until TALLY 1 is performed. Then the program stops.

Parts 3 and 4 illustrate the differences between simple and complex recursions for a situation which involves graphics. Table 3-3 traces the program flow for each of these parts.

Table 3-3. Program flow for Parts 3 and 4.

Part 3 (Simple Recursion)

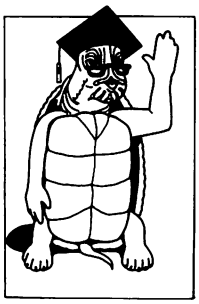
```
CHANGESQUARE    120  draws SQUARE with side = 120
CHANGESQUARE     60  draws SQUARE with side =  60
CHANGESQUARE     30  draws SQUARE with side =  30
CHANGESQUARE     15  draws SQUARE with side =  15
CHANGESQUARE      7  draws SQUARE with side =   7
CHANGESQUARE      3  draws SQUARE with side =   ?
                                                (program terminates)
```

Part 4 (Complex Recursion)

```
CHANGESQUARE    120  calls   CHANGESQUARE     60  (no output)
CHANGESQUARE     60  calls   CHANGESQUARE     30  (no output)
CHANGESQUARE     30  calls   CHANGESQUARE     15  (no output)
CHANGESQUARE     15  calls   CHANGESQUARE      7  (no output)
CHANGESQUARE      7  calls   CHANGESQUARE      3  (no output)
CHANGESQUARE      3  calls   CHANGESQUARE      7  (because of STOP)
CHANGESQUARE      7    draws SQUARE with          side = 7
CHANGESQUARE     15    draws SQUARE with          side = 15
CHANGESQUARE     30    draws SQUARE with          side = 30
CHANGESQUARE     60    draws SQUARE with          side = 60
CHANGESQUARE    120    draws SQUARE with          side = 120
                                                ?(program terminates)
```

Parts 5 and 6 contrast the two types of recursion, but with a more complex graphics design. The reader with a moderate mathematical back-

ground may wish to justify the existence of the various commands and values used to create the design. Otherwise, simply enjoy the effect.



Turtle Truth. Study the following Logo procedures to determine the screen display. Check your answer by entering the commands.

```
TO TREER :RIGHT
IF :RIGHT < 1Ø [STOP]
TREER :RIGHT - 15
SETH 12Ø
FD :RIGHT + 5
SETH 27Ø
FD :RIGHT
END

TO TREEL :LEFT
IF :LEFT < 1Ø [STOP]
SETH 27Ø
FD :LEFT
SETH 6Ø
FD :LEFT + 5
TREEL :LEFT - 15
END

TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP [STOP]
REPEAT :MANY [ FD :SIDE LT 36Ø/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

TO TREE :LEFT :RIGHT :SIDE :STEP :MANY
SETBG 1
SETPC 2
TREEL :LEFT
TREER :RIGHT
PU HOME PD
SETH 9Ø
FD :SIDE/2
SETH 27Ø
SETPC Ø
FILL :SIDE :STEP :MANY
PU HOME PD
HT
END

TREE 6Ø 6Ø 2Ø 1 4
```

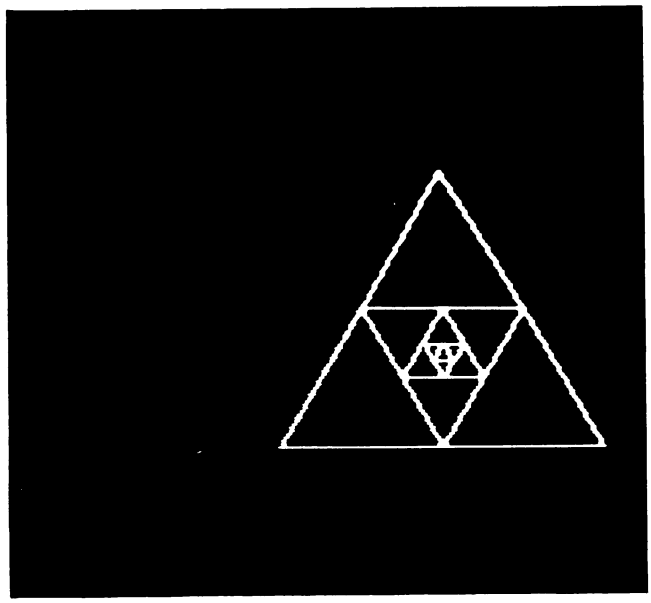


Figure 3-40.

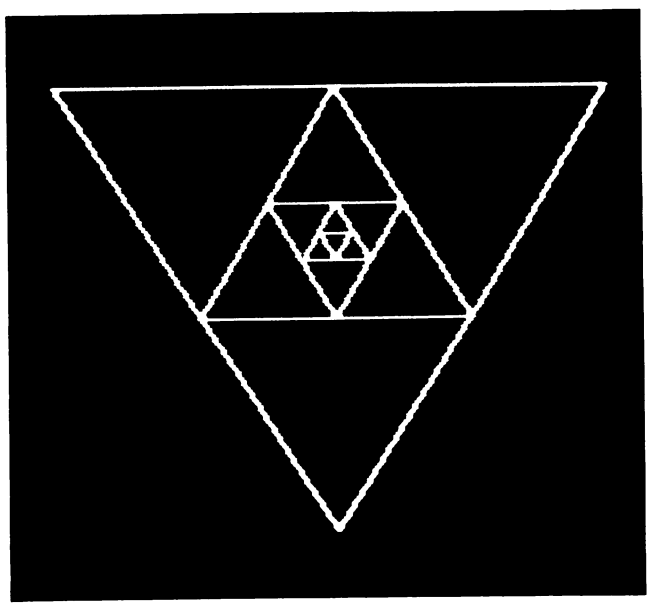


Figure 3-41.

NOTE Save these procedures in a file named RECURSION. Add the following Turtle Tester and Turtle Teaser procedures to the file RECURSION or save them separately under another name.

Turtle Tester. Write a program that uses simple recursion to generate a pattern similar to that which is illustrated in Figure 3-40. (Answer on page 232.)

Turtle Teaser. Write a procedure that uses complex recursion to create a design similar to that in Figure 3-41.

Reference Point 3

Command Review. The following chart contains a summary of all the commands, abbreviations, and definitions introduced in this chapter.

Color Commands

SETPC N Sets the current PenColor to the color assigned by the input number N (0 to 5).

SETBG N Sets the current background screen to the color assigned by the input number N (0 to 6).

PX PENREVERSE causes the turtle to attempt to switch the background color with the PenColor. When the turtle moves it draws where there are no lines and erases where there are previously drawn lines.

SETPEN [WORD N] Sets the pen state to the two inputs enclosed in brackets. WORD must be PU, PD, PE, or PX and N must be the color number desired.

Table 3-4 summarizes all of the possible combinations of drawing colors and background color screens. The interior of the chart describes the resulting effects of these combinations.

Screen Display

FENCE The turtle is restricted to moving only within the dimensions of the screen (240 high and 280 wide). Any attempt to move beyond these values results in the error message TURTLE OUT OF BOUNDS.

WINDOW The turtle is permitted to move beyond the boundaries of the screen and disappear from view. The user is only able to view the portion of the turtle's path drawn within the screen dimensions although the turtle continues to draw beyond that view.

WRAP This is the default or normal mode of screen display. The turtle wraps off one edge of the screen and onto the opposite edge. The turtle is always within the screen boundaries.

Table 3-4. Possible combinations of drawing colors and background color screens.

		Color of Background (SETBG)					
		0 Black	1 White	2 Green	3 Violet	4 Orange	5 Blue
Attempted Color of Drawn Line (SETPC)	0 Black	no	black	black	black	black	black
	1 White	white	no	white	white	white	white
	2 Green	green	green	no	bleed	no	bleed
	3 Violet	violet	violet	bleed	no	bleed	no
	4 Orange	orange	orange	no	bleed	no	bleed
	5 Blue	blue	blue	bleed	no	bleed	no

Arithmetic Operations

x + y Addition is the sum when x is added to y.

x − y Subtraction is the difference when y is subtracted from x.

x * y Multiplication is the product when x is multiplied by y.

x / y Division is the quotient when x is divided by y.

RANDOM x Selects a random non-negative integer between 0 and x − 1, inclusive.

Predicates and Functions

x = y x equals y.

x < y x is less than y.

x > y x is greater than y.

Print Commands

PRINT [. . .] Prints the words or lists contained within the brackets. The abbreviation PR may also be used.

PR PC Prints the current pencolor.

PR BG Prints the current background color.

PR PEN Prints a two-word description of current state of the pen. The first word is PENUP, PENDOWN, PENREVERSE, or PENERASE and the second word is the number of the current pencolor.

PONS Prints out the names of all global variables and their current values.

CLEARTEXT Removes all text from the entire text screen. There is no abbreviation for this command.

Flow of Control

CO Causes a program interrupted by CTRL-Z to continue.

CTRL-W Stops program execution until any key is pressed.

CTRL-Z Causes the program to stop until CO is typed.

IF CONDITION ACTION1 ACTION2 ACTION2 A conditional statement that checks if the condition is true or false. If the condition is true, ACTION1 is executed. If the statement is false, ACTION2 is executed (either on the same line or the next line).

WAIT N Causes the procedure to pause for a unit of time N * ($^1/_{60}$th of a second).

Variables

:NAME Pronounced "dots name," defines NAME to be a local variable. The contents of NAME is assigned as input through a procedure call and is accessible only to procedures which include it in the procedure heading and definition.

MAKE "NAME DATA Defines NAME to be a global variable where DATA is what is assigned to NAME. The contents of NAME are accessible by any procedure.

Application. Enter the following commands and study the resulting picture. Notice you are loading several previously created files and erasing unwanted procedures from each to accumulate desired procedures in the

workspace. You will use these procedures to produce a new picture and save them together in a new file named HOME.SWEET.HOME.

1. Type in the following commands to load the file created in Activity 3.3.

```
LOAD "COLORFUL.STUFF
POTS
```

Erase all the procedures except the three listed below.

```
SQUARE
PANES
RECTANGLE
```

NOTE Remember, to erase a procedure type:

```
ER "FLASH2
```

It is not necessary to type the variables that follow the title.

NOTE Type POTS to verify the existence of the three desired procedures in the workspace. Be careful that you do not erase these procedures as you load the next two files.

2. Enter the following commands to load in the file created in Activity 3.4.

```
LOAD "COLORFUL.POSIES
POTS
```

Erase all the procedures in this file except those listed below. Be careful!

```
POSIE
PETAL
STEM
LEAF
FLOWER
ASSIGN
```

Your workspace should now contain nine procedures.

3. Type in the following commands to load the file created in Activity 3.7.

```
LOAD "SOLID.PICS
POTS
```

Erase all the procedures except those listed below.

```
COLORFILL
HOUSE
MOVE
FILL
GLOBALVALUES
```

4. You should now have the following fourteen procedures in your workspace. To verify that they are correct, compare your procedures to those listed below.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO PANES :SIDE
SQUARE :SIDE/4
SQUARE :SIDE/8
FD :SIDE/8 RT 90
FD :SIDE/8 LT 90
SQUARE :SIDE/8
LT 90 FD :SIDE/8
RT 90 BK :SIDE/8
END

TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH LT 90 FD :WIDTH LT 90]
END

TO FILL :SIDE
IF :SIDE < :STEP [STOP]
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP
END

TO COLORFILL :SIDE
SETPC :COLOR
FILL :SIDE
END

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH 0 PD
END

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
SETPC :COLORHOUSE
GLOBALVALUES 4 :COLORHOUSE 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
SETPC :COLORROOF
GLOBALVALUES 3 :COLORROOF 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END
```

```
TO GLOBALVALUES :FIRST :SECOND :THIRD
MAKE "MANY :FIRST
MAKE "COLOR :SECOND
MAKE "STEP :THIRD
END

TO POSIE
STEM
FLOWER
END

TO ASSIGN
MAKE "COLORSTEM 2
MAKE "COLORFLOWER (1 + RANDOM 5)
END

TO PETAL
REPEAT 2 [ARCR 2Ø 6Ø RT 12Ø]
END

TO FLOWER
SETPC :COLORFLOWER
REPEAT 8 [PETAL RT 36Ø/8]
PU
BK 3Ø
PD
END

TO STEM
SETPC :COLORSTEM
LT 9Ø
LEAF
RT 9Ø
LEAF
SETH Ø
FD 3Ø
END

TO LEAF
REPEAT 2 [ARCR 23 6Ø RT 12Ø]
END
```

5. Edit PANES to read as follows and add the two new procedures.

```
TO PANES :SIDE
GLOBALVALUES 4 1 1
RT 9Ø
COLORFILL :SIDE/4
SETH Ø
SETPC Ø
SQUARE :SIDE/4
SQUARE :SIDE/8
```

```
FD :SIDE/8 RT 9Ø
FD :SIDE/8 LT 9Ø
SQUARE :SIDE/8
LT 9Ø FD :SIDE/8
RT 9Ø BK :SIDE/8
END

TO RAYS
SETPC 4
REPEAT 2Ø [PU FD 2Ø PD FD 1Ø PU BK 3Ø RT 18]
END

TO SUN :SIDE :MANY
SETPC (1 + RANDOM 5)
MOVE 9Ø :MANY
IF :SIDE < 1 [STOP]
CIRCLER :SIDE
SUN :SIDE - :MANY :MANY
END
```

6. Enter the following procedure. Notice it is the accumulation of the procedures now in your workspace. Compare your screen display to the picture illustrated in Figure 3-42.

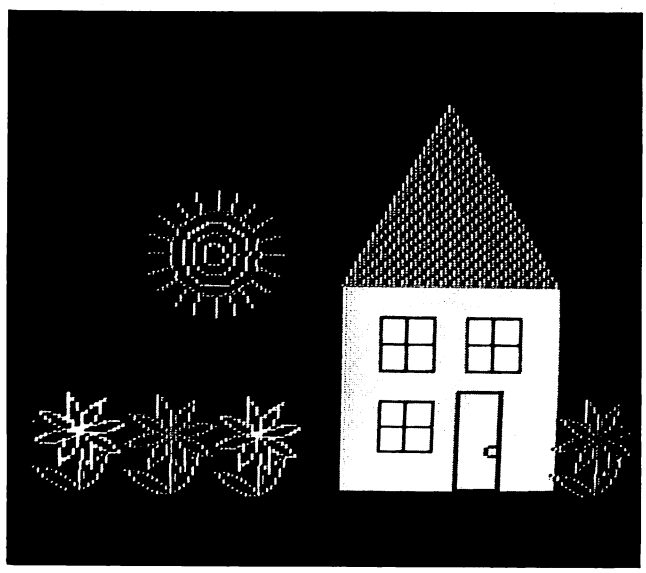


Figure 3-42.

```
TO SCENE
CLEARTEXT
FULLSCREEN
MOVE 18Ø 75
HOUSE 1ØØ 5 4
```

```
MOVE 9Ø 75
SETPC Ø
RECTANGLE 5Ø 2Ø
FD 2Ø
CIRCLEL 3
MOVE 27Ø 57
PANES 1ØØ
MOVE Ø 4Ø
PANES 1ØØ
MOVE 9Ø 4Ø
PANES 1ØØ
MOVE 135 9Ø
REPEAT 4 [ASSIGN POSIE MOVE 9Ø 4Ø]
MOVE 215 15Ø
SUN 2Ø 5
RAYS
PU HOME FD
SPLITSCREEN
PR [HOME, SWEET, HOME]
HT
END

SCENE

SAVE "HOME.SWEET.HOME
```

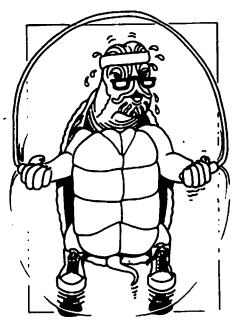
Suggested Projects

1. Add several birds and clouds in the sky.
2. Make the door a different color than the house.
3. Add a chimney to the roof.
4. Make a path leading to the door.
5. Change the color of the house and the roof.

4

Coordinating Turtle Points

Activity 4.1 Turtle Positions



Turtle Trials. This activity introduces the use of (X,Y) Cartesian coordinates.

1. The Logo coordinate system is pictured in Figure 4-1. The Roman numerals define the four quadrants of the Cartesian coordinate system. The purpose of this part is for you to locate various boundary coordinates of the display screen, identified by letters A through I. Enter the following commands and write the ordered pairs in the indicated spaces, both below and on Figure 4-1.

NOTE Cartesian coordinates are not entirely new to you if you have plotted points on coordinate paper (graph paper). You can locate a point on graph paper by giving its distance from two perpendicular lines. Perpendicular lines form right angles (90 degrees) with each other. In Figure 4-1, lines L1 and L2 are perpendicular lines which intersect at point A, or the origin. Points on L1 to the right of A are positive and to the left of A are negative. Similarly, points on L2 above A are positive and below A are negative.

The location of any point in relation to A is identified by an ordered pair (X,Y), where X is the horizontal position in relation to A and Y is the vertical position in relation to A. For example, point W (−10,20) is 10 units to the left of A and 20 units above A. Notice point A (the origin) is identified as (0,0) and is the starting position of the turtle.

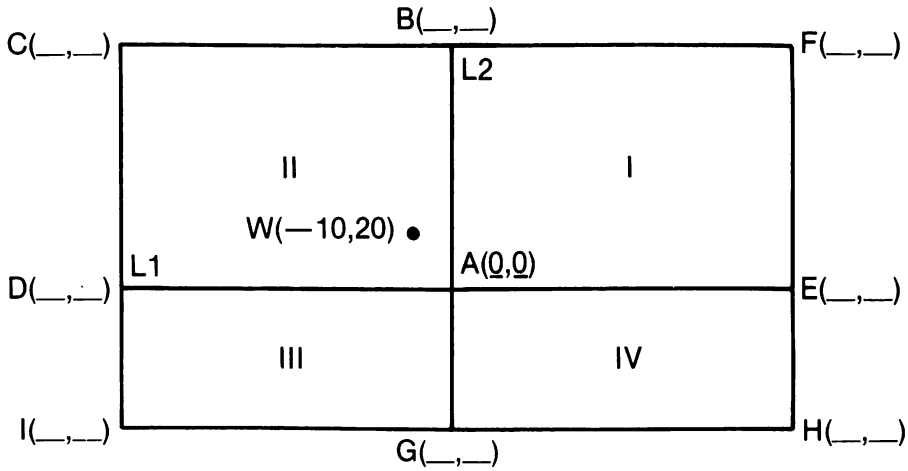


Figure 4-1.

```
CS
PR POSITION    A(Ø,Ø)
FD 12Ø
PR POS         B(—,—)
LT 9Ø
FD 139
PR POS         C(—,—)
LT 9Ø
FD 12Ø
PR POS         D(—,—)
LT 9Ø
FD 278
PR POS         E(—,—)
LT 9Ø
FD 12Ø
PR POS         F(—,—)
LT 9Ø
FD 139
PR POS         B(—,—)
LT 9Ø
FD 12Ø
PR POS         A(—,—)
FD 79
PR POS         G(—,—)
LT 9Ø
FD 139
PR POS         H(—,—)
```

```
LT 18Ø
FD 278
PR POS          I(—,—)
RT 9Ø
FD 79
PR POS          D(—,—)
RT 9Ø
FD 278
PR POS          E(—,—)
RT 9Ø
FD 79
PR POS          H(—,—)
PU HOME PD
PR POS          A(—,—)
```

2. Enter the following commands. Observe the path of the turtle and fill in the information on the lines provided. Compare the coordinates of the points with those obtained in Part 1.

```
CS
PR POS          A(Ø,Ø)
SETY 12Ø
PR POS          B(—,—)
SETX -139
PR POS          C(—,—)
SETY Ø
PR POS          D(—,—)
SETX 139
PR POS          E(—,—)
SETY 12Ø
PR POS          F(—,—)
SETX Ø
PR POS          B(—,—)
SETY Ø
PR POS          A(—,—)
SETY -79
PR POS          G(—,—)
SETX 139
PR POS          H(—,—)
SETX -139
PR POS          I(—,—)
SETY Ø
PR POS          D(—,—)
SETX 139
PR POS          E(—,—)
SETY -79
PR POS          H(—,—)
PU HOME PD
PR POS          A(—,—)
```

NOTE The coordinates of the points in Part 1 are the same as those in Part 2.

3. Enter the following commands. Record your observations on the lines following the commands. Compare the coordinates of the points with those in Parts 1 and 2.

```
CS
PR POS    A(—,—)
SETPOS [Ø 12Ø]
PR POS    B(—,—)
SETPOS [-139 12Ø]
PR POS    C(—,—)
SETPOS [-139 Ø]
PR POS    D(—,—)
SETPOS [139 Ø]
PR POS    E(—,—)
SETPOS [139 12Ø]
PR POS    F(—,—)
SETPOS [Ø 12Ø]
PR POS    B(—,—)
SETPOS [Ø Ø]
PR POS    A(—,—)
SETPOS [Ø -79]
PR POS    G(—,—)
SETPOS [139 -79]
PR POS    H(—,—)
PR XCOR _____
PR YCOR _____
SETPOS [-139 -79]
PR POS    I(—,—)
PR XCOR _____
PR YCOR _____
SETPOS [-139 Ø]
PR POS    D(—,—)
PR XCOR _____
PR YCOR _____
SETPOS [139 Ø]
PR POS    E(—,—)
SETPOS [139 -79]
PR POS    H(—,—)
PU HOME PD
PR POS    A(—,—)
```

Are the coordinates of points in Part 3 the same as those in Parts 1 and 2? _____

4. Type the following commands and compare the screen display with the illustration in Figure 4-2.

```
CS
PU SETPOS [-1ØØ 1ØØ] PD
SETPOS [1ØØ 1ØØ]
SETPOS [1ØØ -7Ø]
SETPOS [-1ØØ -7Ø]
SETPOS [-1ØØ 1ØØ]
SETPOS [1ØØ -7Ø]
SETPOS [-1ØØ -7Ø]
SETPOS [1ØØ 1ØØ]
```

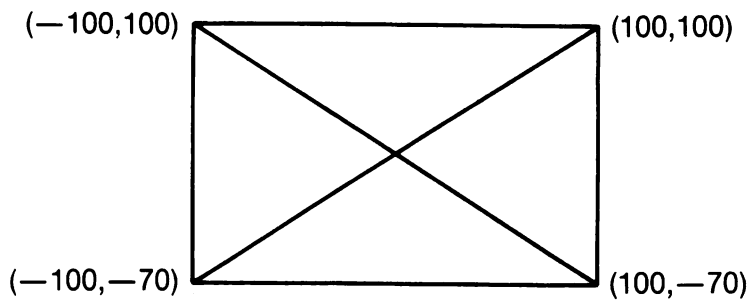


Figure 4-2.

5. Enter the recursive procedures listed below and describe the screen display on the line provided.

```
CS
TO EXPLOSION :NUMBER
HT
IF :NUMBER = Ø [STOP]
PU HOME PD
MAKE "SIGN RANDOM 2
IF :SIGN = Ø [MAKE "SIGN -1]
MAKE "XPOS (RANDOM 12Ø)*:SIGN
MAKE "YPOS RANDOM 1ØØ
SETPOS [:XPOS :YPOS]
EXPLOSION :NUMBER - 1
END

EXPLOSION 7Ø _____
```

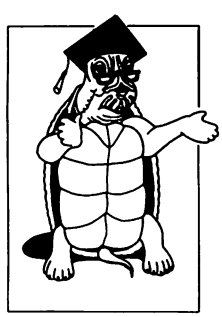
6. Edit EXPLOSION to read as follows and describe the screen displays on the lines provided.

```
CS
TO EXPLOSION :NUMBER
HT
IF :NUMBER = Ø [STOP]
PU HOME PD
```

```
MAKE "SIGN RANDOM 2
IF :SIGN = Ø [MAKE "SIGN -1]
MAKE "XPOS (RANDOM 12Ø)*:SIGN
MAKE "YPOS RANDOM 1ØØ
SETPOS SENTENCE :XPOS: YPOS
EXPLOSION :NUMBER - 1
END

EXPLOSION 7Ø _____
CS
EXPLOSION 1ØØ _____
```

7. Edit EXPLOSION to read as follows, then describe the screen displays on the lines provided.

```
CS
TO EXPLOSION :NUMBER
HT
IF :NUMBER = Ø [STOP]
PU HOME PD
MAKE "SIGN RANDOM 2
IF :SIGN = Ø [MAKE "SIGN -1]
MAKE "XPOS (RANDOM 12Ø)*:SIGN
MAKE "YPOS RANDOM 1ØØ
SETPOS LIST :XPOS :YPOS
EXPLOSION :NUMBER - 1
END

EXPLOSION 7Ø _____
CS
EXPLOSION 1ØØ _____
```



Turtle Talk. Apple Logo permits the turtle to be positioned anywhere on the screen by using Cartesian coordinates. Figure 4-3 summarizes the screen coordinates that should have been obtained during the activity. Each coordinate is defined as an ordered pair, where the first number indicates the horizontal position (X) and the second number indicates the vertical position (Y). Both are required to identify each point uniquely in the two-dimensional space in relation to the origin. Notice the center of the screen is defined to be (0,0). Positive and negative numbers are used to indicate the other positions using the coordinate system typically used in mathematics.

The coordinates of the position of the turtle may be observed at any time by using PRINT POSITION (abbreviated PR POS). The first number indicates the horizontal position (or X coordinate) and the second number identifies the vertical position (or Y coordinate). If the value of only the X

coordinate is desired, PR XCOR will print this value. PR YCOR prints only the value of the Y coordinate.

There are several methods by which the turtle can be located on the screen. First, the FD and BK commands used in Part 1 serve as review. In Part 2, SETX and SETY position the turtle in either a horizontal or vertical direction. SETPOS, which is introduced in Part 3, places the turtle at the coordinates of a position that is immediately obvious by the values included in the command. For example, SETPOS [100 50] moves the turtle to a point where the (X,Y) coordinate is (100,50).

A difficulty encountered in Part 5 is that SETPOS requires a list of two numbers as input but will not accept variables as inputs. Parts 6 and 7 resolve the difficulty by using the SENTENCE and LIST commands, respectively. Both commands take a collection of inputs and form a list. Since SENTENCE usually deals with textual material, the LIST command will be used throughout the remainder of the book to create a list of input variables for use with the SETPOS command.

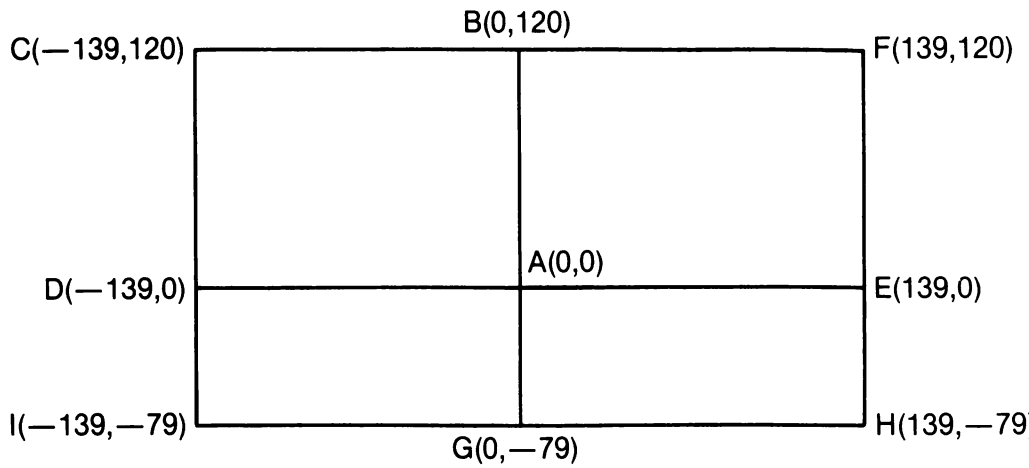
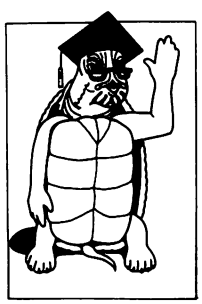


Figure 4-3.

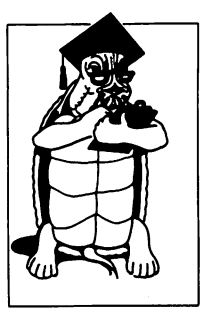


Turtle Truth. Study the procedures and inputs and predict their outcome. Type in the commands and observe the results.

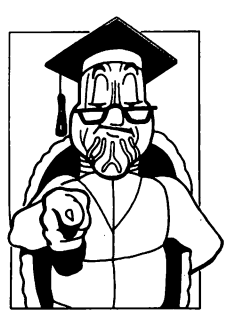
```
CS
TO AXIS :XPOS :YPOS
```

```
LONG :YPOS
LAT : XPOS
END

TO LONG :YPOS
PU
SETX Ø
SETY :YPOS
PD FD :YPOS*-2
END

TO LAT :XPOS
PU
SETX :XPOS
SETY Ø
SETH 9Ø
PD FD :XPOS*-2
END

TO MOVE :XPOS :YPOS :ANGLE
PU
SETPOS LIST :XPOS :YPOS
SETH :ANGLE
PD
END

TO LBRACKET :SIDE :XPOS :YPOS :ANGLE
MOVE :XPOS :YPOS :ANGLE
FD :SIDE
SETH Ø
FD :SIDE*2
SETH 9Ø
FD :SIDE
END

TO RBRACKET :SIDE :XPOS :YPOS :ANGLE
MOVE :XPOS :YPOS :ANGLE
FD :SIDE
SETH Ø
FD :SIDE*2
SETH 27Ø
FD :SIDE
END

TO POSITIVE :SIDE
PD REPEAT 4 [FD :SIDE BK :SIDE RT 9Ø]
END

TO NEGATIVE :SIDE
PD
SETH 9Ø
FD :SIDE*2
END
```

```
AXIS -6Ø -6Ø
CS
AXIS -139 -12Ø
LBRACKET 1Ø 3Ø 4Ø 27Ø
RBRACKET 1Ø 8Ø 4Ø 9Ø
MOVE 4Ø 5Ø Ø
POSITIVE 1Ø
MOVE 65 5Ø Ø
POSITIVE 1Ø
```

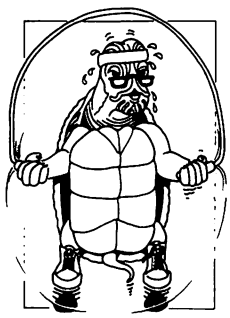


Turtle Tester. Use the procedures provided in the Turtle Truth to label each quadrant of the coordinate axis system with the appropriate positive or negative symbols indicating the correct coordinate signs. One quadrant has been done for you in the Turtle Truth. Save your work under the file name GRID. (Answer on page 233.)



Turtle Teaser. Use the procedures LBRACKET, RBRACKET, POSITIVE, and NEGATIVE to produce one master procedure that draws and labels all of the quadrants.

Activity 4.2 Turtle Dots



Turtle Trials. This activity introduces the DOT and TOWARDS commands and demonstrates their use in drawing.

1. Enter the following procedure and record the resulting screen displays on the lines provided.

```
TO LONGS :XPOS :YPOS :LENGTH
IF :XPOS > :LENGTH/2 [STOP]
PU
SETPOS LIST :XPOS :YPOS
PD FD :LENGTH
LONGS :XPOS + 1Ø :YPOS :LENGTH
END

LONGS -11Ø -11Ø 22Ø _____
CS
LONGS -5Ø -5Ø 1ØØ _____
CS
LONGS -4Ø -4Ø 8Ø _____
CS
```

2. Enter the following procedure and record the resulting screen displays on the lines provided.

```
TO LATS :XPOS :YPOS :LENGTH
IF :YPOS > :LENGTH/2 [STOP]
PU
SETH 9Ø
SETPOS LIST :XPOS :YPOS
PD FD :LENGTH
LATS :XPOS :YPOS + 1Ø :LENGTH
END

LATS -11Ø -11Ø 22Ø _____
CS
LATS -8Ø -8Ø 16Ø _____
CS
```

3. Enter the following procedures and describe the results on the lines provided.

```
TO AXIS :LENGTH
PU HOME PD
SETPC 3
REPEAT 4 [FD :LENGTH/2 BK :LENGTH/2 RT 90]
END

TO GRID :XPOS :YPOS :LENGTH
FULLSCREEN
MAKE "FIRSTPEN PEN
LONGS :XPOS :YPOS :LENGTH
LATS :XPOS :YPOS :LENGTH
AXIS :LENGTH
SETPEN :FIRSTPEN
END

GRID -90 -90 180 _____
CS
GRID -50 -50 100 _____
CS
GRID -20 -20 60 _____
CS
GRID 20 20 40 _____
CS
GRID -110 110 220 _____
```

4. Using the GRID that remains on your screen, observe and record the results of the following commands. Draw the results on the grid paper illustrated in Figure 4-4. Notice the action of the turtle as you push return after the SETH TOWARDS command.

```
CTRL-S
SETPC 5
DOT [20 20] _____
SETH TOWARDS [90 20] _____
PR POS _____
PR TOWARDS [90 20] _____

DOT [-100 80] _____
PR POS _____
SETH TOWARDS [-100 80] _____
PR TOWARDS [-100 80] _____

DOT [60 -60] _____
SETH TOWARDS [60 -60] _____
PR TOWARDS [60 -60] _____
```

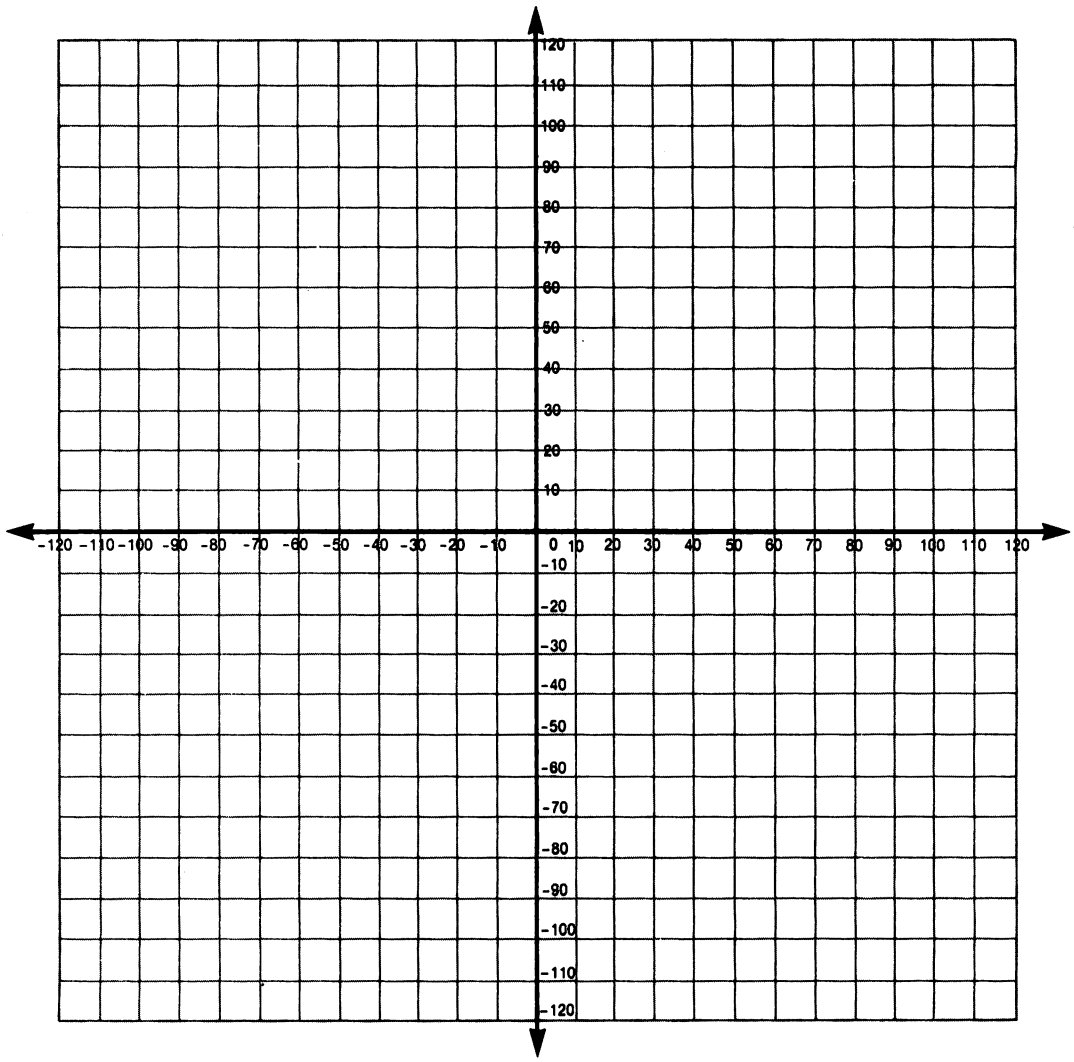


Figure 4-4.

```
DOT [-7Ø -2Ø] _____
SETH TOWARDS [-7Ø -2Ø] _____
PR TOWARDS [-7Ø -2Ø] _____
CS
```

5. Enter the following commands and record the results.

```
SETPC 2
PU SETPOS [-8Ø -6Ø] _____
PD SETPOS [-2Ø 2Ø] _____
SETPOS [2Ø 2Ø] _____
SETPOS [-8Ø -6Ø] _____
HT
```

```
SETPC Ø
DOT [2Ø 2Ø] _____
DOT [-2Ø 2Ø] _____
DOT [-8Ø -6Ø] _____
CS
```

6. Enter the following commands and record the coordinates on the lines provided. Notice how SETH TOWARDS is helpful in creating an irregular shape.

```
SETPC 1
PU
SETPOS [46 2Ø]
SETH TOWARDS [55 25]
PD
FD 12
PR POS _____
SETH TOWARDS [58 28]
FD 8
PR POS _____
SETH TOWARDS [68 4Ø]
FD 5
PR POS _____
SETH TOWARDS [7Ø 45]
FD 5
PR POS _____
SETH TOWARDS [78 46]
FD 5
PR POS _____
SETH TOWARDS [8Ø 46]
FD 5
PR POS _____
SETH TOWARDS [85 43]
FD 5
PR POS _____
SETH TOWARDS [86 42]
FD 3
PR POS _____
SETH TOWARDS [87 4Ø]
FD 5
PR POS _____
SETH TOWARDS [88 36]
```

```
FD 5
PR POS _____________________________________
SETH TOWARDS [8Ø 3Ø]
FD 5
PR POS _____________________________________
SETH TOWARDS [75 3Ø]
FD 5
PR POS _____________________________________
SETH TOWARDS [6Ø 1Ø]
FD 8
PR POS _____________________________________
SETH TOWARDS [5Ø 1Ø]
FD 8
PR POS _____________________________________
SETH TOWARDS [4Ø 5]
FD 5
PR POS _____________________________________
SETH TOWARDS [45 Ø]
FD 5
PR POS _____________________________________
SETH TOWARDS [5Ø -1Ø]
FD 1Ø
PR POS _____________________________________
HT
DOT [78 4Ø]
PU HOME PD
```

7. The above example shows how the output of PR POS or PR TOWARDS might be used to assist in the drawing of irregular shapes in immediate mode. WORM suggests how the values can be rounded and used as coordinates for SETPOS to create the image as a procedure.

```
TO WORM
SETPC 2
PU
SETPOS [46 2Ø]
SETH TOWARDS [55 25]
PD
FD 12
SETPOS [56 26]
SETPOS [61 32]
SETPOS [64 36]
SETPOS [67 4Ø]
SETPOS [72 43]
SETPOS [76 44]
```

```
SETPOS [81 44]
SETPOS [84 43]
SETPOS [88 39]
SETPOS [88 34]
SETPOS [84 32]
SETPOS [79 31]
SETPOS [74 25]
SETPOS [67 21]
SETPOS [62 18]
SETPOS [59 15]
SETPOS [56 5]
HT
DOT [78 40]
PU HOME PD
END

WORM ____________________________________
```

8. Enter the following commands in edit mode. The sequence of commands will demonstrate the use of CTRL-K and CTRL-Y to facilitate the typing of repeated commands.

```
EDIT "LEAF
```

When the screen display changes to the editor, you should see TO LEAF ■ at the top. Press return and continue to enter the commands as listed.

```
SETPC 2
PU
SETPOS [CTRL-A CTRL-K CTRL-Y 0 60]
PD
CTRL-Y 5 70]
CTRL-Y 10 77]
CTRL-Y 15 80]
CTRL-Y 20 82]
CTRL-Y 25 90]
CTRL-Y 25 80]
CTRL-Y 22 75]
CTRL-Y 20 70]
CTRL-Y 15 65]
CTRL-Y 10 60]
CTRL-Y 0 60]
END

LEAF ____________________________________
```

NOTE As you have seen, it is often worthwhile to define a procedure in edit mode rather than definition mode. Edit mode allows you more flexibility for correcting typing errors and permits you to use "yank" to reduce repetitive typing. Remember that CTRL-Y yanks back the most

recently "killed" line. Do not use CTRL-K on any other lines while using this technique or you will lose the command from the edit buffer.

9. Observe the diagram illustrated in Figure 4-5. Notice how the coordinates to be plotted are obtained. Use the CTRL-K and CTRL-Y technique demonstrated in Part 6 to enter the following procedure.

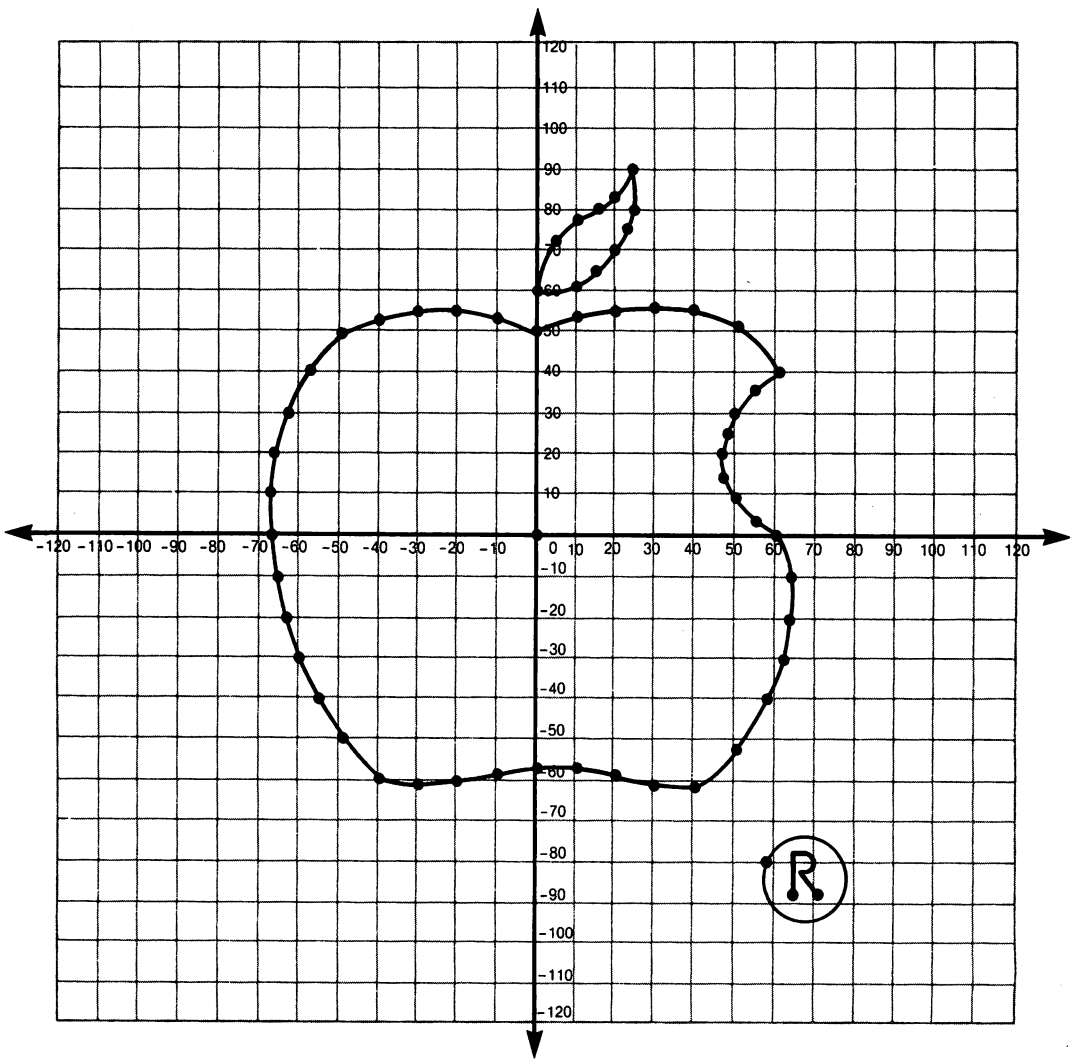
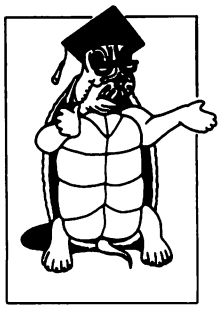


Figure 4-5.

```
EDIT "APPLE

TO APPLE
SETPC 3
```

```
PU
SETPOS [Ø 5Ø]
PD
SETPOS [1Ø 52]
SETPOS [2Ø 55]
SETPOS [3Ø 57]
SETPOS [4Ø 55]
SETPOS [5Ø 5Ø]
SETPOS [6Ø 4Ø]
SETPOS [55 35]
SETPOS [5Ø 3Ø]
SETPOS [47 25]
SETPOS [46 2Ø]
SETPOS [47 15]
SETPOS [5Ø 1Ø]
SETPOS [55 5]
SETPOS [6Ø Ø]
SETPOS [64 -1Ø]
SETPOS [63 -2Ø]
SETPOS [62 -3Ø]
SETPOS [58 -4Ø]
SETPOS [5Ø -53]
SETPOS [4Ø -61]
SETPOS [3Ø -6Ø]
SETPOS [2Ø -59]
SETPOS [1Ø -57]
SETPOS [Ø -57]
SETPOS [-1Ø -59]
SETPOS [-2Ø -6Ø]
SETPOS [-3Ø -61]
SETPOS [-4Ø -6Ø]
SETPOS [-5Ø -5Ø]
SETPOS [-55 -4Ø]
SETPOS [-6Ø -3Ø]
SETPOS [-62 -2Ø]
SETPOS [-65 -1Ø]
SETPOS [-67 Ø]
SETPOS [-67 1Ø]
SETPOS [-65 2Ø]
SETPOS [-62 3Ø]
SETPOS [-57 4Ø]
SETPOS [-5Ø 5Ø]
SETPOS [-4Ø 52]
SETPOS [-3Ø 55]
SETPOS [-2Ø 55]
SETPOS [-1Ø 52]
SETPOS [Ø 5Ø]
END

APPLE _____

LEAF _____
```

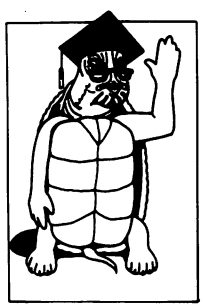


Turtle Talk. Parts 1, 2, and 3 develop three procedures that are used in GRID to draw pictures of different size graph paper with centers located at (0,0). LONGS draws the vertical lines, LATS draws the horizontal lines, and AXIS draws the colored X,Y axis. The horizontal and vertical lines are each ten dots (or pixels) apart. A pixel is the smallest graphic image that can be placed on the screen. The three input values for GRID determine the starting X-coordinate, Y-coordinate, and the size of the graph paper, respectively. The grid is used to introduce the DOT and TOWARDS commands in Part 4.

The DOT command places a dot or pixel on the display screen at a designated coordinate position. For example, DOT [80 −10] displays a dot in the lower right section of the screen, where the X-coordinate is 80 and the Y-coordinate is −10. Part 5 combines the SETPOS and DOT commands to draw a triangle, then remove its vertices.

The SETH TOWARDS command turns the turtle so it faces toward a designated point on the display screen. The heading of the newly directed turtle is revealed by the PR TOWARDS command. Part 6 demonstrates the combination of two commands used to draw more intricate designs at a trial/error level. Part 7 rounds the numeric values obtained by the PR POS command to define coordinates for the SETPOS command in the procedure WORM.

Parts 8 and 9 illustrate how CTRL-K and CTRL-Y are used effectively for entering repetitive data while in edit mode.



Turtle Truth. Enter the following procedures and notice the use of PU, PD, and other primitives within the procedures. Study the additions that will create an elaborate APPLE.DEMO.

```
TO STRIPES
PU
SETPOS [-57 40]
PD
SETPC 2
SETPOS [60 40]
PU
SETPOS [-65 20]
PD
SETPC 3
```

```
SETPOS [45 2Ø]
PU
SETPOS [-67 Ø]
PD
SETPC 4
SETPOS [-62 -2Ø]
SETPC 5
SETPOS [61 -2Ø]
PU
SETPOS [-55 -4Ø]
PD
SETPC 2
SETPOS [52 -4Ø]
END

TO TRADEMARK
SETPC 4
PU
SETPOS [55 -8Ø]
PD
CIRCLER 1Ø
PU
SETPOS [65 -85]
PD
SETPC 1
SETPOS [65 -75]
SETH 9Ø
ARCR 3 18Ø
SETPOS [7Ø -85]
END

TO DISPLAY
FULLSCREEN
APPLE
WORM
LEAF
STRIPES
TRADEMARK
PU HOME PD
HT
END

DISPLAY _____
CTRL-S

SAVE "APPLE.DEMO

ERPS
```

Turtle Tester. Study the illustration in Figure 4-6 and write a procedure using SETPOS and DOT to create the flag. Use color and fill procedures wherever you desire. Save the procedures under the file name of your choice. Use the ERPS command before attempting the Turtle Teaser. (Answer on page 234.)

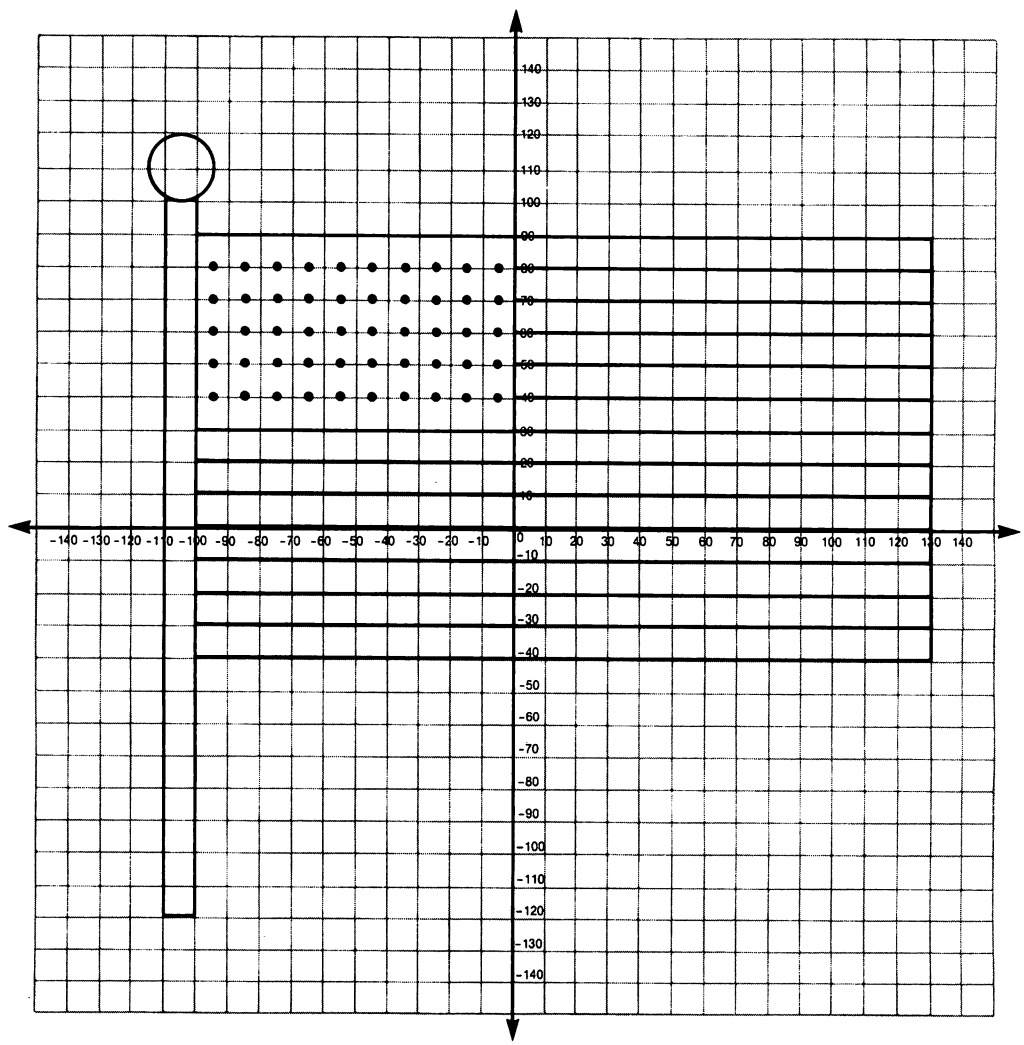


Figure 4-6.

Turtle Teaser. Use the graph paper in Figure 4-7 to sketch a picture of your choice, then define the procedures necessary to draw that picture. Save the procedures under a file name of your choice.

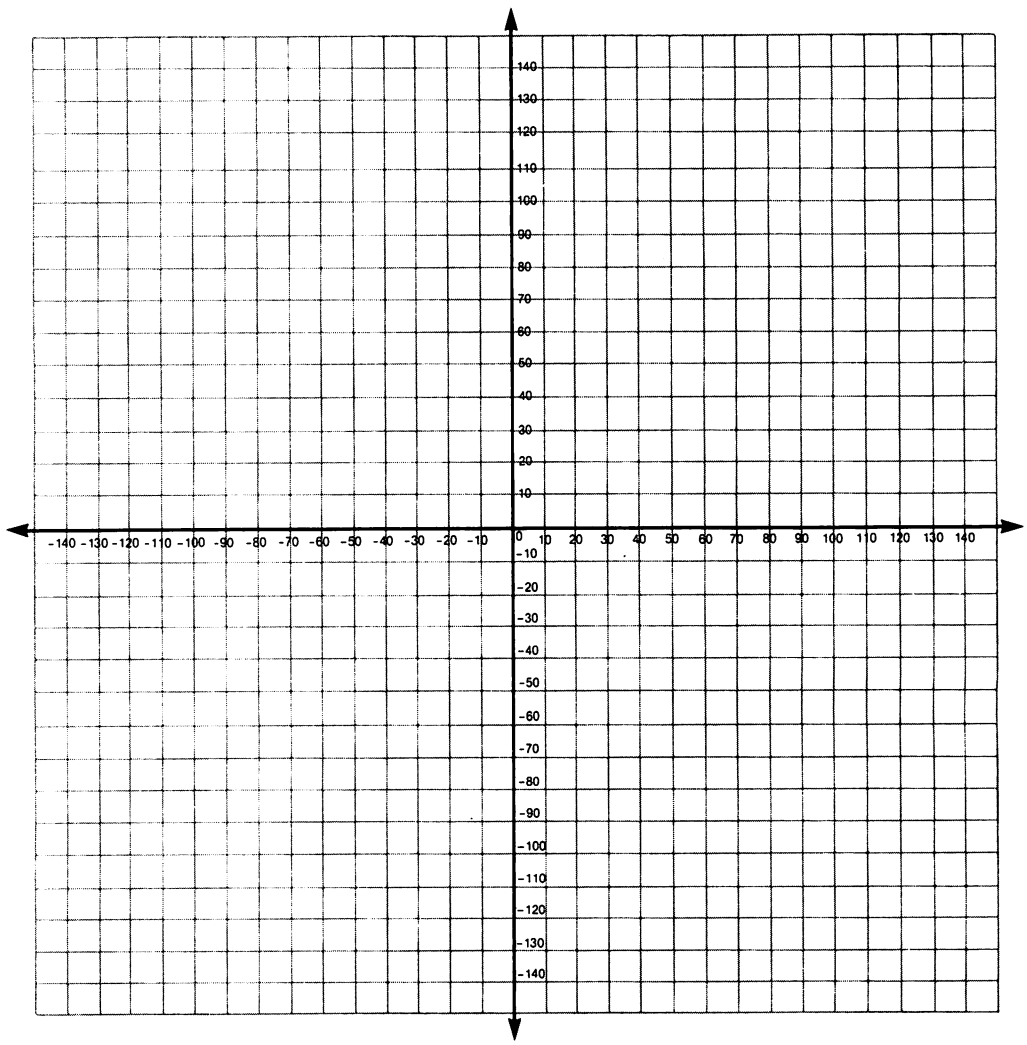
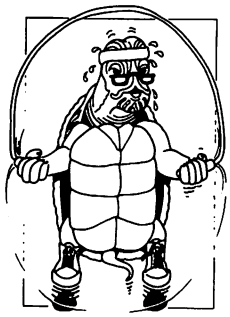


Figure 4-7.

Activity 4.3 Turtle Placement



Turtle Trials. This activity incorporates variables in procedures to draw a diagram anywhere on the screen.

1. Enter the following procedure to draw the frame of a kite as illustrated in Figure 4-8. Observe the drawing on the screen.

```
TO FRAME1
SPLITSCREEN
PU
SETPOS [50 40]
PD SETPC 5
SETPOS [50 90]
SETPOS [30 70]
SETPOS [50 40]
SETPOS [70 70]
SETPOS [50 90]
SETPOS [70 70]
SETPOS [30 70]
PU
SETPOS [50 40]
PD
END
FRAME1
```

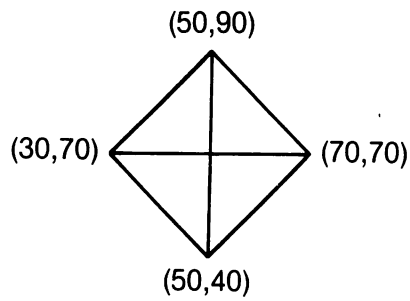


Figure 4-8.

2. Create FRAME2 by editing FRAME1 to read as listed below.

```
TO FRAME2
SPLITSCREEN
```

```
PU
SETPOS [-6Ø -3Ø]
PD SETPC 5
SETPOS [-6Ø 2Ø]
SETPOS [-8Ø Ø]
SETPOS [-6Ø -3Ø]
SETPOS [-4Ø Ø]
SETPOS [-6Ø -2Ø]
SETPOS [-4Ø Ø]
SETPOS [-8Ø 3Ø]
PU
SETPOS [-6Ø -3Ø]
PD
END
```

FRAME1 _____

FRAME2 _____

Compare the shape and position of the kite drawn in FRAME1 with the kite drawn in FRAME2. _____

Describe how the coordinates used in each procedure are similar. _____

_____

3. Create FRAME by editing FRAME1 so it read as follows. Figure 4-9 identifies the new form of (X,Y) coordinates.

```
TO FRAME :XPOS :YPOS
FULLSCREEN
PU
SETPOS LIST :XPOS :YPOS :COLORFRAME
PD SETPC :COLORFRAME
SETPOS LIST :XPOS :YPOS + 5Ø
SETPOS LIST :XPOS - 2Ø :YPOS + 3Ø
SETPOS LIST :XPOS :YPOS
SETPOS LIST :XPOS  + 2Ø :YPOS + 3Ø
SETPOS LIST :XPOS :YPOS + 5Ø
```

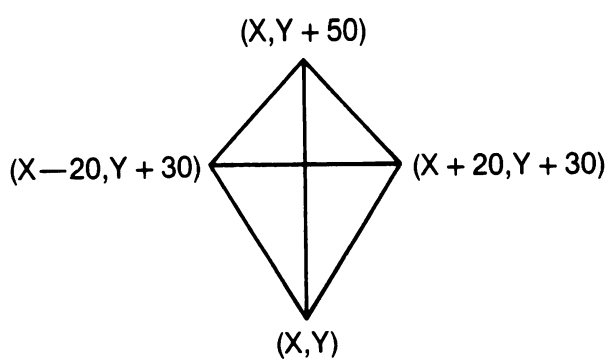


Figure 4-9.

```
SETPOS LIST :XPOS + 20 :YPOS + 30
SETPOS LIST :XPOS - 20 :YPOS + 30
PU
SETPOS LIST :XPOS :YPOS
SETPC 1
END

FRAME 50 40 5 _____
FRAME -60 -30 3 _____
```

4. Create FRAME3 by editing FRAME once more so it reads as follows. Note the change of (X,Y) coordinates to agree with the illustration in Figure 4-10.

```
TO FRAME3 :XPOS :YPOS :COLORFRAME
FULLSCREEN
PU
SETPOS LIST :XPOS :YPOS - 50
PD SETPC :COLORFRAME
SETPOS LIST :XPOS :YPOS
SETPOS LIST :XPOS - 20 :YPOS - 20
SETPOS LIST :XPOS :YPOS - 50
SETPOS LIST :XPOS + 20 :YPOS - 20
SETPOS LIST :XPOS :YPOS
SETPOS LIST :XPOS + 20 :YPOS - 20
SETPOS LIST :XPOS - 20 :YPOS - 20
PU
SETPOS LIST :XPOS :YPOS - 50
SETPC1
END

FRAME3 50 40 5 _____

FRAME -85 50 4 _____

FRAME3 -2 -15 3 _____

FRAME3 -75 -50 2 _____
```

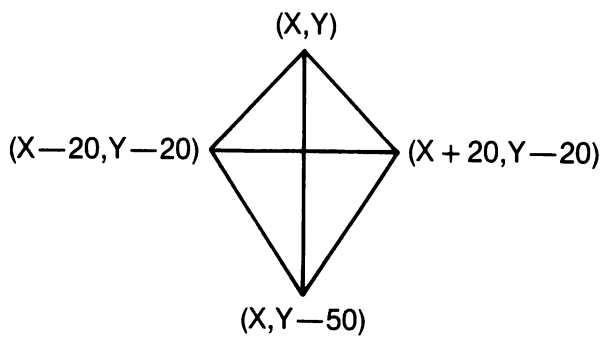
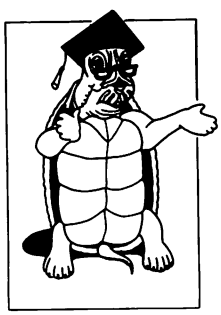


Figure 4-10.

5. Enter the following procedure to draw a tail for the kite.

```
FRAME 50 40 5

TO TAIL :POS :YPOS :COLORTAIL
PD
SETPC :COLORTAIL
SETH TOWARD LIST :XPOS - 10 :YPOS - 10
FD 16
SETH TOWARD LIST :XPOS - 16 :YPOS - 20
FD 8
SETH TOWARD LIST :XPOS - 14 :YPOS - 20
FD 8
SETH TOWARD LIST :XPOS - 7 :YPOS - 27
FD 16
SETH TOWARD LIST :XPOS + 24 :YPOS - 49
FD 16
END

FRAME 50 40 5

TAIL 50 40 3
```



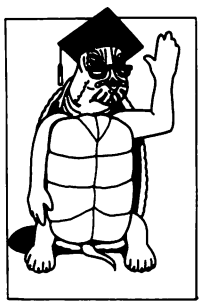
Turtle Talk. Parts 1 and 2 each draw a picture of the frame of a kite, but at different locations on the screen. The procedures FRAME1 and FRAME2 differ only in the coordinates for the SETPOS command. Fortunately, there is a better way to display a diagram in a variety of locations on the screen other than this cumbersome approach.

Part 3 introduces a method whereby the coordinates for the SETPOS command are generalized by using variables, rather than specific numerical values. Figure 4-9 shows that the (X,Y) coordinate is located at the bottom of the kite frame. Other coordinate values are determined by their position in relation to the (X,Y) coordinate. The position of the displayed kite on the screen depends upon the values given to X and Y.

For example, if X = 50 and Y = 40, the bottom of the kite is placed at coordinate (50, 40) on the screen. An (X,Y) coordinate of (−60,−30) positions the bottom of the kite on the screen where X = −60 and Y = −30.

Part 4, which refers to Figure 4-10, demonstrates that the selected position for the (X,Y) coordinate (top of the kite frame) is arbitrary, assuming the other coordinates are modified accordingly.

Part 5 embellishes the kite by adding a tail. Notice the (X,Y) coordinate of TAIL (refer to Figure 4-11) is defined at the top of the tail to assure that FRAME and TAIL are properly coordinated to appear as one connected picture.



Turtle Truth. Enter the procedures as listed below. Study the inputs and predict the position of the kite as determined by the inputs for each variable.

```
TO NIGHT.KITE :NUMBER :XPLACE :YPLACE
 :COLORFRAME :COLORTAIL
STARS :NUMBER
FRAME :XPLACE :YPLACE :COLORFRAME
TAIL :XPLACE :YPLACE :COLORTAIL
BOW
END


TO STARS :NUMBER
IF :NUMBER = Ø [STOP]
FULLSCREEN
HT
MAKE "SIGN RANDOM 2
IF :SIGN = Ø [MAKE "SIGN -1]
MAKE "XPOS (RANDOM 139)*:SIGN
MAKE "SIGN RANDOM 2
IF :SIGN = Ø [MAKE "SIGN -1]
MAKE "YPOS (RANDOM 12Ø)*:SIGN
DOT LIST :XPOS :YPOS
STARS :NUMBER - 1
END


TO BOW
SETPC 1
RT 45
REPEAT 3 [FD 8 RT 12Ø]
LT 9Ø
REPEAT 3 [FD 8 LT 12Ø]
END


NIGHT.KITE 5Ø 5Ø 4Ø 3 2 _____
CS
NIGHT.KITE 1ØØ -8Ø -3Ø 2 4 _____
CS
```

Turtle Tester. Create a procedure to draw a kite containing two bows on the tail as illustrated in Figure 4-11. Now write a procedure to draw three different colored kites in the sky. Save it as a file called SKYFULL.OF.KITES. (Answer on page 263.)

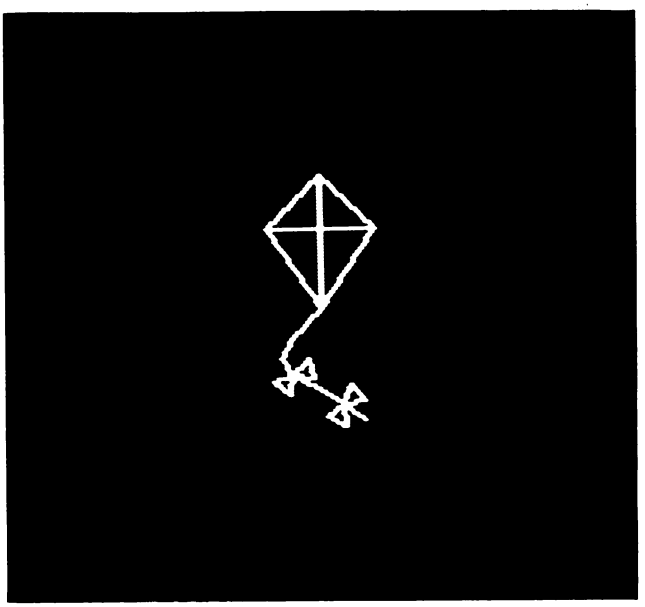
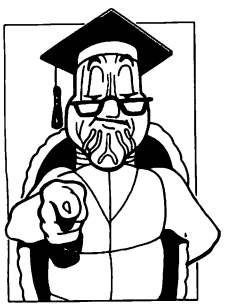
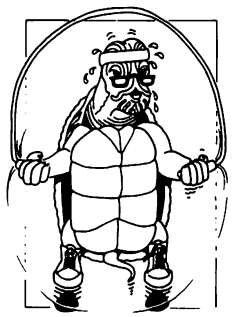


Figure 4-11.



Turtle Teaser. Design a box kite using the commands introduced in Activities 4.1 and 4.2. It may be helpful to draw the design and plot the coordinates on graph paper before working in toplevel mode or defining the procedure itself.

Activity 4.4 Turtle Tessellations



Turtle Trials. This activity demonstrates the development of fullscreen tessellations.

1. Enter the following commands and observe the pattern that is created.

```
CS
TO SQUARE :SIDE
MAKE "COLOR (1 + RANDOM 5)
SETPC :COLOR
REPEAT 4 [FD :SIDE RT 90]
FD :SIDE
END

PR 240/30 _____

REPEAT 8 [SQUARE 30] _____

CS
PR 240/19 _____

REPEAT 12.6316 [SQUARE 19] _____
CS

PR 240/20 _____
REPEAT 12 [SQUARE 20] _____
CS
```

2. Enter the procedures below and observe the results.

```
TO VROW :LENGTH :XPLACE :YPLACE
FULLSCREEN
PU SETPOS :XPLACE :YPLACE PD
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 [MAKE "NUMBER :NUMBER*-1]
REPEAT :NUMBER [SQUARE :LENGTH]
END

VROW 30 -140 -120 _____
CS
```

```
VROW 19 -139 -119 _____________________________
CS

VROW 7 -13Ø -11Ø _____________________________
CS
```

3. Edit the procedure to read as follows and observe the results.

```
TO VROW :LENGTH :XPLACE :YPLACE
FULLSCREEN
PU SETPOS :XPLACE :YPLACE PD
MAKE "FIRSTPOS POS
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < Ø [MAKE "NUMBER :NUMBER*-1]
REPEAT :NUMBER [SQUARE :LENGTH]
PU SETPOS :FIRSTPOS PD
END

VROW 3Ø -14Ø -12Ø _____________________________
CS

VROW 14 -132 -115 _____________________________
CS

VROW 45 -139 -119 _____________________________
CS
```

4. Enter the following procedure and observe the screen display.

```
TO MOVE :LENGTH
PU RT 9Ø FD :LENGTH LT 9Ø PD
END
VROW 45 -139 -119 _____________________________
MOVE 45 _______________________________________
VROW 45 -139 + 45 -119 ________________________
CS
```

5. Enter the following procedure and record you observations.

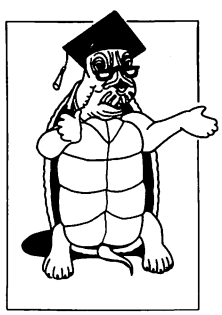
```
TO ACROSS :LENGTH :XPLACE :YPLACE
VROW :LENGTH :XPLACE :YPLACE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE
END

ACROSS 45 -139 -119 ___________________________
```

Use CTRL-G to stop the procedure when it begins to wrap around.

```
CS
```

6. Modify the ACROSS procedure to read as follows and observe the results.

```
TO ACROSS :LENGTH :XPLACE :YPLACE
IF :XPLACE > 140 - :LENGTH [STOP]
VROW :LENGTH :XPLACE :YPLACE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE
END

ACROSS 45 -139 -119 _____________________
CS

ACROSS 2Ø -13Ø -11Ø _____________________
CS

ACROSS 1Ø -135 -115 _____________________
CS
SAVE "FIRST.TESS
```



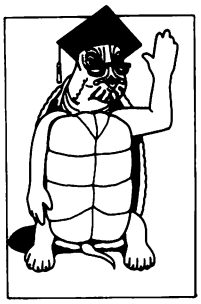
Turtle Talk. A tessellation is a careful arrangement of parts into a coherent pattern or picture. Tessellations are of particular interest to mathematicians because they demonstrate the geometric relationship of mathematics to art. Since tessellations essentially fill space in a repetitive manner, the computer becomes a useful tool for the creation of tessellations.

The first step is to decide upon some design to be tessellated. To begin, this activity tessellates a simple square, as defined in Part 1.

With Turtle Graphics, tessellations are two-dimensional designs. Therefore, the repetitive drawing of the diagram must be controlled vertically and horizontally, and be symmetrically coordinated to form a mosaic-type design.

Parts 2 and 3 develop a procedure to control the vertical positioning of the design. Once the column design is completed, conditions must be established for drawing in an adjacent column. The MOVE procedure defined in Part 4 accomplishes this task. ACROSS controls the horizontal drawing of the tessellation, to avoid screen wrap-around.

The Turtle Truth example tessellates a more complex design, our previously created turtle. Several of the procedures used for tessellating the square are reused to tessellate the turtle. The original turtle is redefined using variables to control its size. The turtle design is assigned a square area of space, then tessellated in much the same way as a square of that size.



Turtle Truth. Enter the following procedure to create a new tessellation. Notice the object used in the tessellation is contained in a SQUARE, and can be plugged into the VROW and ACROSS procedures to create a new tessellation. Compare your final screen display to the illustration in Figure 4-12.

```
TO TURTLE :DISTANCE
START :DISTANCE
SHELL :DISTANCE
SHAPE :DISTANCE
PAWS :DISTANCE
TAIL :DISTANCE
HEAD :DISTANCE
RETURN :DISTANCE
END

TO START :DISTANCE
PU RT 90 FD :DISTANCE*1.5 LT 90 PD
END

TO BOX :DISTANCE
REPEAT 4 [FD :DISTANCE RT 90]
END

TO SHELL :DISTANCE
REPEAT 2 [BOX :DISTANCE FD :DISTANCE]
BOX :DISTANCE
BK :DISTANCE
LT 90
BOX :DISTANCE
BK :DISTANCE*2
BOX :DISTANCE
END

TO SHAPE :DISTANCE
RT 90
REPEAT 4 [FD :DISTANCE LT 45 FD :DISTANCE*1.4 LT 45]
LT 90
END

TO PAWS :DISTANCE
LT 45
FD :DISTANCE*.4
LT 90
REPEAT 4 [ARCR :DISTANCE/3 180 LT 90
FD :DISTANCE*.4 RT 45 FD :DISTANCE RT 45 FD :DISTANCE*.4 LT 90]
END
```

```
TO TAIL :DISTANCE
RT 90
FD :DISTANCE
RT 45
FD :DISTANCE*.4
TRI :DISTANCE/3 120
END

TO TRI :DISTANCE :ANGLE
REPEAT 3 [FD :DISTANCE LT :ANGLE]
END

TO HEAD :DISTANCE
BK :DISTANCE*4
RT 90
FD: DISTANCE*3
LT 90
REPEAT 3 [FD :DISTANCE RT 120]
END

TO RETURN :DISTANCE
FD :DISTANCE
RT 90
BK :DISTANCE*3
PU LT 90 FD :DISTANCE*1.5 RT 90 PD
END

TURTLE 10

SQUARE 4*10
```

Edit the following procedures MOVE, VROW, and ACROSS to read as follows:

```
TO MOVE :LENGTH
PU FD :LENGTH PD
END

TO VROW :LENGTH :XPLACE :YPLACE :DISTANCE
FULLSCREEN
PU SETPOS LIST :XPLACE :YPLACE PD
MAKE "FIRSTPOS POS
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 [MAKE "NUMBER :NUMBER*-1]
REPEAT :NUMBER [TURTLE :DISTANCE MOVE :LENGTH]
PU SETPOS :FIRSTPOS PD
END

VROW 40 -139 -119 10
CS

TO ACROSS :LENGTH :XPLACE :YPLACE :DISTANCE
IF :XPLACE > 140 - :LENGTH [STOP]
```

```
VROW :LENGTH :XPLACE :YPLACE :DISTANCE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + LENGTH :YPLACE :DISTANCE
HT
END

ACROSS 4Ø -14Ø -12Ø 1Ø
```

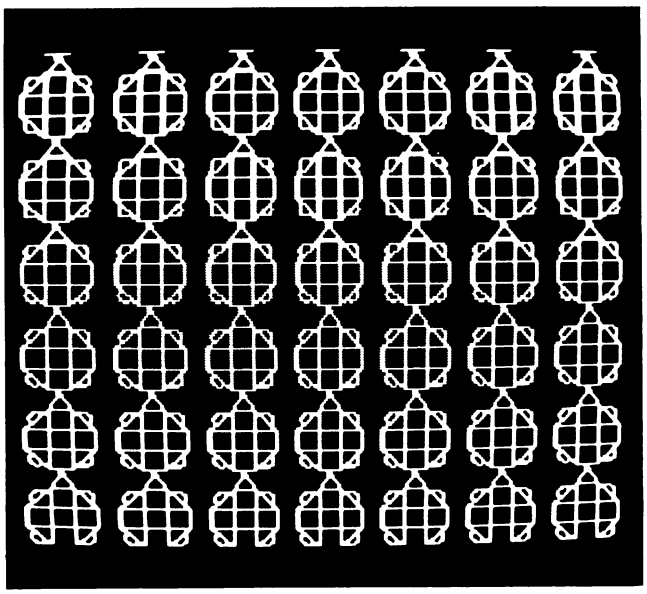


Figure 4-12.

```
SAVE "TURTLE.TESS
CS
```

Turtle Tester. Study the illustration in Figure 4-13 and enter the following procedures used to create STAR. Modify the VROW, MOVE, and ACROSS procedures to create a fullscreen tessellation with STAR. (Answer on page 237.)

```
TO STAR :DISTANCE :MANY
COLOR
REPEAT :MANY [VARICUBE :DISTANCE RT 36Ø/:MANY]
END

TO COLOR
SETPC (1 + RANDOM 5)
END
```

```
TO SIDE :DISTANCE
RT 45 FD :DISTANCE
LT 45 FD :DISTANCE
LT 135 FD :DISTANCE
LT 45 FD :DISTANCE
RT 180
END

TO VARICUBE :DISTANCE
BOX :DISTANCE
SIDE :DISTANCE
RT 90 FD :DISTANCE LT 90
SIDE :DISTANCE
LT 90 FD :DISTANCE RT 135 FD :DISTANCE LT 45
BOX :DISTANCE
RT 45 BK :DISTANCE LT 45
END

STAR 10 8
```

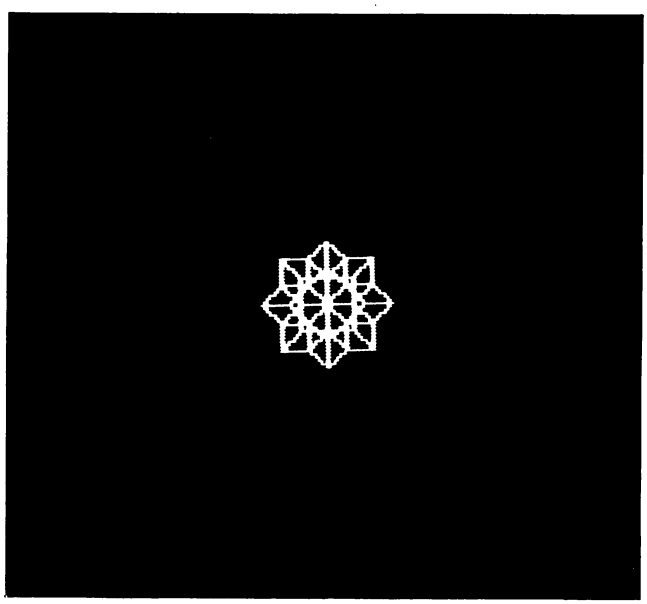


Figure 4-13.

NOTE After you have successfully tessellated STAR, save all the necessary procedures in a file named STAR.TESS.

Turtle Teaser. Create your own object and use it in a fullscreen tessellation. Save this tessellation under a file name of your choice.

Reference Point 4

Command Review. This section introduces the definitions of terms and commands presented in Chapter 4.

Terms

Screen coordinates The Turtle Graphics screen is divided into small entities called dots or pixels. There are approximately 278 columns and 240 rows of pixels. Each pixel has a unique identification known as a screen coordinate. Figure 4-14 identifies several screen coordinates and indicates their relative position on the display screen.

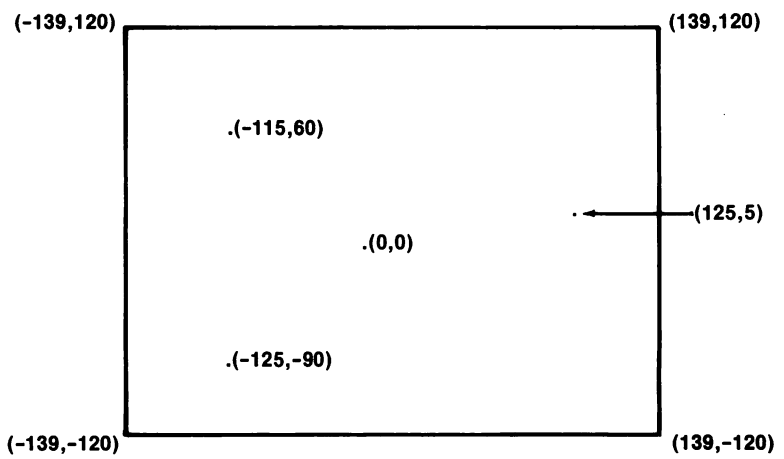


Figure 4-14.

Ordered pairs Each screen coordinate is described as an ordered pair, where the first number indicates the horizontal position (X) and the second number indicates the vertical position (Y). The ordered pair (0,0) is located at the center of the screen. Positive and negative numbers are used in the ordered pairs to indicate the relative position of the point to

(0,0). It is common to refer to an ordered pair as an (X,Y) coordinate where X means horizontal direction and Y means vertical direction.

Tessellation A computer is particularly useful when repetitive processes are required. A tessellation is a design that results from repeatedly drawing a picture in a coordinated and consistent manner to form a mosaic-like pattern.

Commands

SETPOS [N1 N2] Moves the turtle to the screen coordinate identified by the ordered pair (N1,N2) where N1 and N2 are integer constants. For example:

```
SETPOS [-5Ø 4Ø]
```

moves the turtle to coordinate (−50,40).

SETPOS LIST :X :Y Moves the turtle to the screen coordinate identified by the ordered pair (X,Y) where X and Y are integer variables. Input values for X and Y must be provided prior to executing the command. For example:

```
TO MOVE :X :Y
SETPOS LIST :X :Y
END

MOVE -50 4Ø
```

moves the turtle to coordinate (−50,40).

SETX N1 Moves the turtle horizontally to the X-coordinate specified by N1, where N1 is an integer input constant. For example:

```
SETPOS [-5Ø 4Ø]
SETX 5Ø
```

moves the turtle from coordinate (−50,40) to (50,40).

SETY N2 Moves the turtle vertically to the Y-coordinate specified by N2, where N2 is an integer input constant. For example:

```
SETPOS [-5Ø 4Ø]
SETY -6Ø
```

moves the turtle from coordinate (−50,40) to (−50,-60).

PR XCOR Prints the current X-coordinate (or horizontal position) of the turtle on the screen. For example:

```
SETPOS [-5Ø 4Ø]
PR XCOR
```

prints −50 on the screen.

PR YCOR Prints the current Y-coordinate (or vertical position) of the turtle on the screen. For example:

```
SETPOS [-50 40]
PR YCOR
```

prints 40 on the screen.

PR POS Prints the current (X,Y) coordinate screen position of the turtle. For example:

```
SETPOS [-50 40]
PR POS
```

prints −50. 40. on the screen.

SETH TOWARDS [N1 N2] Points the tip of the turtle to the screen coordinate (N1,N2), where N1 and N2 are input integer constants. For example:

```
SETH TOWARDS [-100 50]
```

points the turtle towards the screen coordinate (−100,50).

PR TOWARDS [N1 N2] Prints the heading from the turtle's current position to point (N1,N2). For example:

```
SETPOS [0 0]
SETH TOWARDS [139 0]
PR TOWARDS [139 0]
```

prints 90. on the screen.

Application. Enter the following procedures and study the resulting display. Identify the procedures controlling the border and the positioning of the message. Compare your final screen display to the illustration in Figure 4-15.

```
TO SECTION :LENGTH :STEP :MANY :DISTANCE
MAKE "FIRSTPOS POS
IF :LENGTH < :STEP [STOP]
REPEAT :MANY [FD :LENGTH RT 360/:MANY
SECTION :LENGTH - :STEP :STEP :MANY :DISTANCE
SETPOS :FIRSTPOS
PU FD :DISTANCE PD
END

TO HROW :LENGTH :STEP :MANY :XPOS :YPOS :DISTANCE
PU SETPOS LIST :XPOS :YPOS PD
MAKE "FIRSTPOS POS
MAKE "NUMBER (2*XPOS)/:LENGTH
IF :NUMBER < 0 [MAKE "NUMBER :NUMBER*-1]
REPEAT :NUMBER [SECTION :LENGTH :STEP :MANY :DISTANCE]
PU SETPOS :FIRSTPOS PD
END

TO VROW :LENGTH :STEP :MANY :XPOS :YPOS :DISTANCE
```

```
PU SETPOS LIST :XPOS :YPOS PD
MAKE "FIRSTPOS POS
MAKE "NUMBER (2*YPOS)/:LENGTH
IF :NUMBER < Ø [MAKE "NUMBER :NUMBER*-1]
REPEAT :NUMBER [SECTION :LENGTH :STEP :MANY :DISTANCE]
PU SETPOS :FIRSTPOS PD
END

TO FRAME
SETBG 1
SETPC 5
VROW 15 1.5 4 -139 -119 15
FD 15 RT 9Ø FD 15
HROW 15 1.5 4 -139 119 15
FD 15 RT 9Ø FD 15
VROW 15 1.5 4 139 119 15
FD 15 RT 9Ø FD 15
HROW 15 1.5 4 139 -119 15
HT
END

TO SAMPLER
FULLSCREEN
HT
FRAME
H -8Ø 4Ø 5 7 3
O -4Ø 4Ø 4 7 3
M Ø 4Ø 2 7
E 4Ø 4Ø 3 7 4 3
S -1ØØ -2Ø 3 2 5
W -6Ø -2Ø 3 7
E -2Ø -2Ø 5 7 4 3
E 2Ø -2Ø 4 7 4 3
T 6Ø -2Ø 2 6 5
H -8Ø -8Ø 4 7 3
O -4Ø -8Ø 3 7 3
M 0 -8Ø 2 7
E 4Ø -8Ø 5 7 4 3
END

TO W :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
CROSS :XPLACE + 6 :YPLACE + :COLOR
CROSS :XPLACE + 12 :YPLACE + 12 :COLOR
CROSS :XPLACE + 18 :YPLACE + 6 :COLOR
END

TO M :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
CROSS :XPLACE + 6 :YPLACE + 3Ø :COLOR
```

```
CROSS :XPLACE + 12 :YPLACE + 24 :COLOR
CROSS :XPLACE + 18 :YPLACE + 30 :COLOR
END

TO S :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
HSIDEN2 :XPLACE :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE :YPLACE + 18 :COLOR :LENGTH2
HSIDEN2 :XPLACE :YPLACE + 36 :COLOR :LENGTH2
VSIDEN1 :XPLACE :YPLACE + 24 :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE + 6 :COLOR :LENGTH1
END

TO T :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE + 12 :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE :YPLACE + 36 :COLOR :LENGTH2
END

TO E :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2 :LENGTH3
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE + 6 :YPLACE + 18 :COLOR :LENGTH3
HSIDEN2 :XPLACE + 6 :YPLACE + 36 :COLOR :LENGTH2
END

TO VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
IF :LENGTH1 = 0 [STOP]
CROSS :XPLACE :YPLACE :COLOR
VSIDE7 :XPLACE :YPLACE + 6 :COLOR :LENGTH1 - 1
END

TO HSIDEN2 :XPLACE :YPLACE :COLOR :LENGTH2
IF :LENGTH2 = 0 [STOP]
CROSS :XPLACE :YPLACE :COLOR
HSIDE3 :XPLACE + 6 :YPLACE :COLOR :LENGTH2 - 1
END

TO O :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE + 6 :YPLACE + 36 :COLOR :LENGTH2
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
END

TO H :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE + 18 :COLOR :LENGTH2
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
END

TO HSIDE3 :XPLACE :YPLACE :COLOR :LENGTH2
IF :LENGTH2 = 0 [STOP]
CROSS :XPLACE :YPLACE :COLOR
```

```
HSIDE3 :XPLACE + 6 :YPLACE :COLOR :LENGTH2 - 1
END

TO VSIDE7 :XPLACE :YPLACE :COLOR :LENGTH1
IF :LENGTH1 = Ø[STOP]
CROSS :XPLACE :YPLACE :COLOR
VSIDE7 :XPLACE :YPLACE + 6 :COLOR :LENGTH1 - 1
END

TO CROSS :XPLACE :YPLACE :COLOR
SETPC :COLOR
PU SETPOS LIST :XPLACE :YPLACE
PD
SETPOS LIST :XPLACE + 5 :YPLACE + 5
PU SETPOS LIST :XPLACE + 5 :YPLACE
PD
SETPOS LIST :XPLACE :YPLACE + 5
END

SAMPLER
SAVE "DOMESTIC
```

Figure 4-15.

Suggested Projects

1. Change the color of the background or letters used in DOMESTIC.
2. Retrieve KITE from Activity 4.3 and use it with FRAME created in Reference Point 4 to design your own cross stitch sampler.

3. Design a tessellation to use as a frame for your own cross stitch sampler or the one provided in DOMESTIC.
4. Change the procedure CROSS to a design that resembles a French knot or some other embroidery stitch.
5. Change the message in the sampler to read "ONE CHAPTER TO GO" or some other pithy statement.

5

Creating Turtle Temptations

Introduction

Logo is designed to be a tool for developing thinking skills. This book concentrates on one component of Logo, Turtle Graphics, as a medium by which intellectual growth is stimulated. Through the control of a "turtle," the computer provides an environment in which problem solving processes can occur. Exploration and learning by discovery are necessary ingredients in a Logo learning experience. CHAPTERS ONE through FOUR introduced you to the elements of Turtle Graphics with this theme in mind.

You were first asked to explore and play with various Logo commands in the TURTLE TRIALS section. Some results were successful; some were not. You learned correct syntax (language and punctuation) and observed the resulting screen display patterns created by these commands.

Following this experience, you were presented with a TURTLE TRUTH which suggested how these various commands could be combined to display a more sophisticated idea. It is the creating and solving of more sophisticated ideas, based upon the successful resolving of related ideas, that becomes the ultimate goal in the Logo learning situation. Let us explore this concept more fully.

The learner has an idea (or graphic design) in mind. Once having established a problem to be solved, the idea is explored by instructing the computerized turtle to draw a picture on the screen. The resulting image either satisfies the learner or does not. Choices, decisions, and additional experimenting to find an acceptable solution are the steps in this problem-solving process.

Once an acceptable solution has been found, the sequence of commands are grouped together to form a procedure and given a name. This process

continues, and the learner develops a collection of procedures which can be used at any time in a new learning sequence. Learning becomes a gradual process of trial and error experiences, where new hypotheses are formed and tested until new resolutions are found.

This individually created language provides an opportunity for unlimited creativity and personal mental activity and development. To nurture this development, learners should choose, design, and implement projects.

Two different approaches can be used to develop these projects: part-to-whole programming and whole-to-part programming. The purpose of Chapter 5 is to present examples of these methods and to provide a list of suggested projects to attempt.

Individual learning styles will influence the approach that the reader selects to use when developing an original project. Both whole-to-part and part-to-whole methods are equally satisfactory.

Part-To-Whole Programming

Part-to-whole programming is an approach where the learner does not have a clearly defined image or design in mind when beginning a project. The programming begins with a basic idea which, upon its successful completion, suggests an elaboration or additional idea. This process continues until no new ideas are conceived and the final project is completed. The following example demonstrates this problem-solving approach.

Let us assume we wish to draw a picture of a computer terminal. Initially, our idea coincides with the diagram illustrated in Figure 5-1. Enter the procedure COMPUTER to obtain the diagram.

```
TO COMPUTER
PU
SETPOS [-80 -60]
PD
SETPOS [-50 -30]
SETPOS [-50 70]
SETPOS [-30 80]
SETPOS [70 80]
SETPOS [50 70]
SETPOS [-50 70]
SETPOS [50 70]
SETPOS [50 -30]
SETPOS [-50 -30]
SETPOS [50 -30]
SETPOS [20 -60]
SETPOS [-80 -60]
SETPOS [-80 -70]
SETPOS [20 -70]
```

```
SETPOS [7Ø -4Ø]
SETPOS [7Ø 8Ø]
PU
SETPOS [-4Ø -2Ø]
PD
SETPOS [-4Ø 6Ø]
SETPOS [4Ø 6Ø]
SETPOS [4Ø -2Ø]
SETPOS [-4Ø -2Ø]
END
```

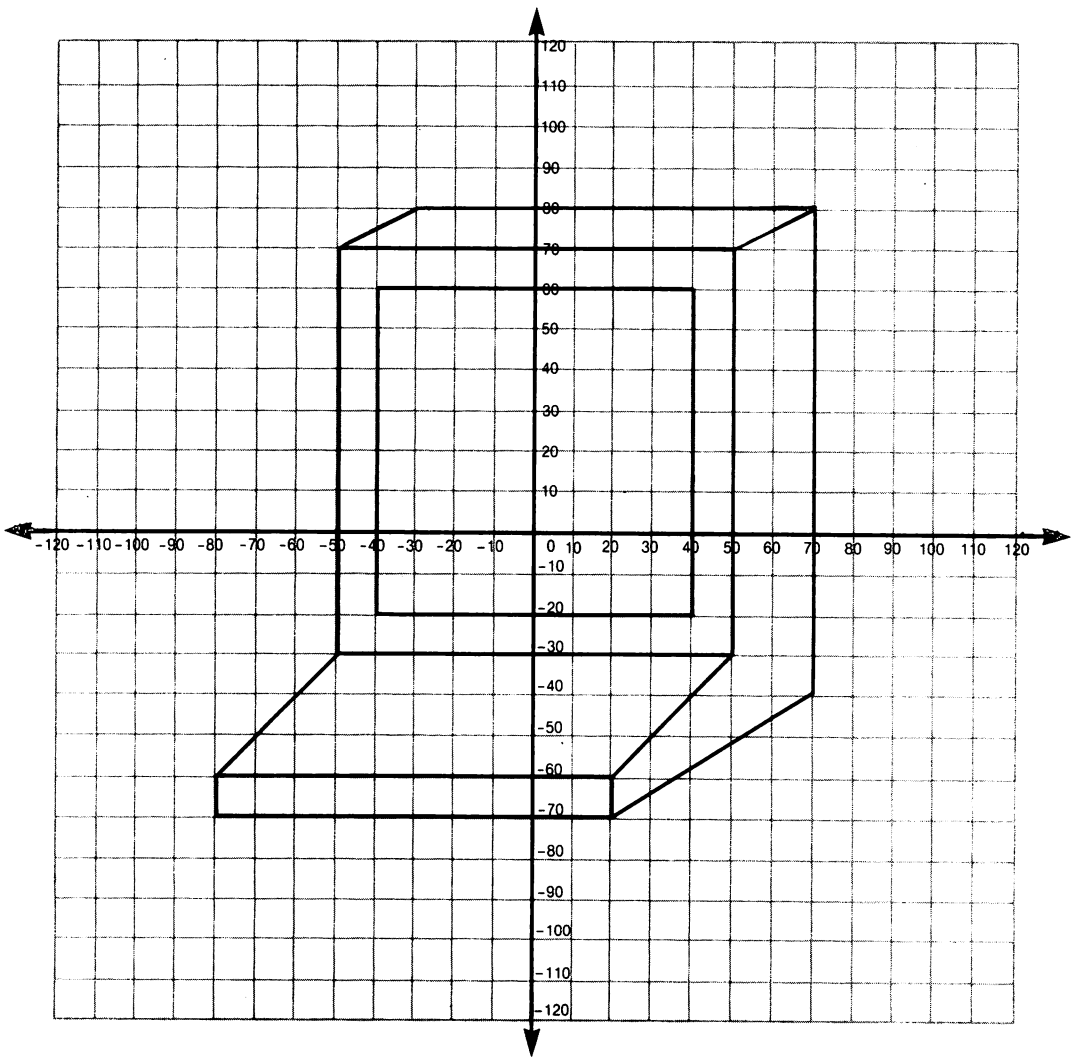


Figure 5-1.

As you observe the computer, you decide its appearance could be improved if the screen were shaded in green. You recall the FILL procedure developed in Activity 3.7 and write a procedure called SCREEN that uses the procedure called FILL. Enter these procedures.

```
TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP [STOP]
REPEAT :MANY [FD :SIDE RT 360/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

TO SCREEN
SETPC 2
FILL 80 1 4
END

SETPC 2
FILL 80 1 4
END
```

You define a procedure named ALL to observe the composite picture as it develops.

```
TO ALL
COMPUTER
SCREEN
END
```

Type ALL and compare your screen display to the illustration in Figure 5-2.

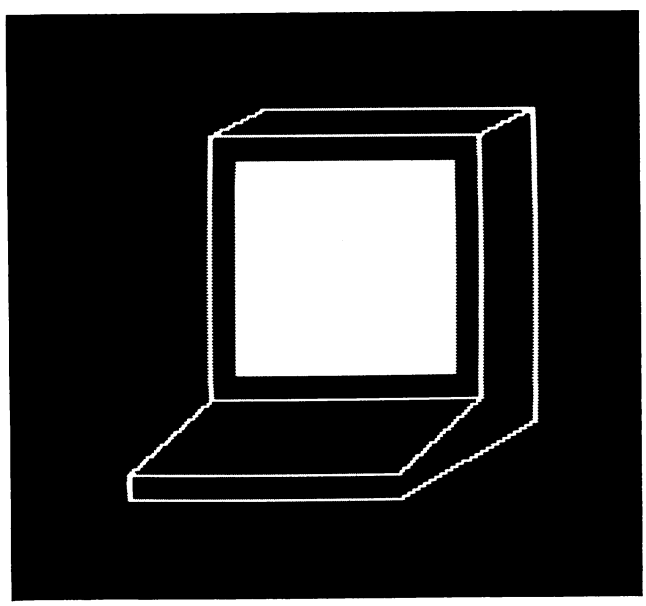


Figure 5-2.

Now that you are satisfied with this diagram, you notice the keyboard portion of the computer is missing keys. You proceed to display keys by

preparing a procedure called KEYS, a ROW procedure to be used in KEYS, then editing ALL to include the procedures.

```
TO KEYS :XPOS :YPOS: TALLYKEY :TALLYROW
SETPC 3
ROW :XPOS :YPOS :TALLYKEY
IF :TALLYROW = 3 [STOP]
KEYS :XPOS - 5 :YPOS - 5 :TALLYKEY :TALLYROW + 1
END

TO ROW :XPOS :YPOS: TALLYKEY
DOT LIST :XPOS :YPOS
IF :TALLYKEY = 9 [STOP]
ROW :XPOS + 10 :YPOS :TALLYKEY + 1
END

TO ALL
COMPUTER
SCREEN
KEYS -5Ø -4Ø Ø Ø
END
```

By typing ALL again, your new diagram should agree with that illustrated in Figure 5-3.

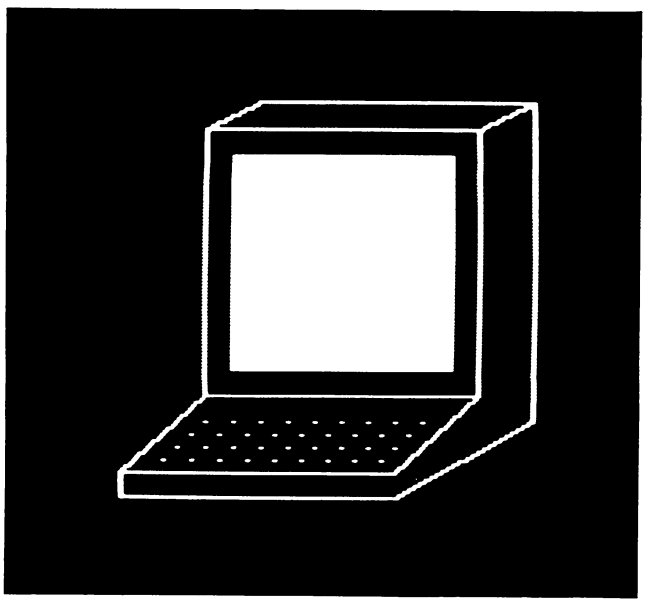


Figure 5-3.

Still not satisfied with the complexity of your creation, you cleverly decide to create a maze on the screen. This is accomplished by adding the next procedure and modifying ALL to read as follows:

```
TO MAZE :LENGTH
SETPC Ø
FD :LENGTH
RT 9Ø
IF :LENGTH < Ø [STOP]
MAZE :LENGTH - 5
END

TO ALL
HT
COMPUTER
SCREEN
KEYS -5Ø -4Ø Ø Ø
PU SETPOS [-3Ø -1Ø] PD
MAZE 6Ø
END
```

Figure 5-4 illustrates the new composite diagram that has been created. Type ALL to observe the diagram on your screen display.

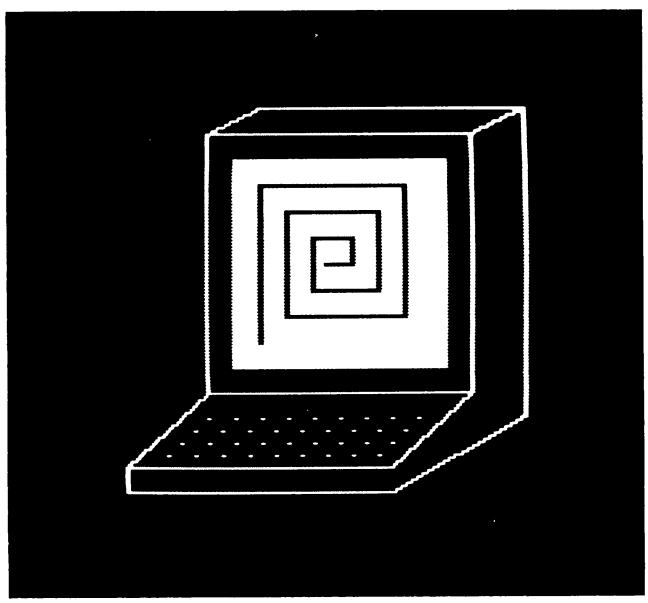


Figure 5-4.

Your creativity is really coming alive! You decide to move the turtle through the maze, as if a racing game were being played. This new idea is accomplished by adding MAZE2 and again modifying ALL.

```
TO MAZE2 :LENGTH
FD :LENGTH
WAIT 2Ø
RT 9Ø
WAIT 2Ø
```

```
IF :LENGTH < Ø [STOP]
MAZE2 :LENGTH - 5
END

TO ALL
HT
COMPUTER
SCREEN
KEYS -5Ø -4Ø Ø Ø
PU SETPOS [-3Ø -1Ø] PD
MAZE 6Ø
ST
PU SETPOS [-25 -5] SETH Ø
MAZE2 5Ø
SETH Ø
PD
END
```

By typing ALL, you will observe the diagram illustrated in Figure 5-5.

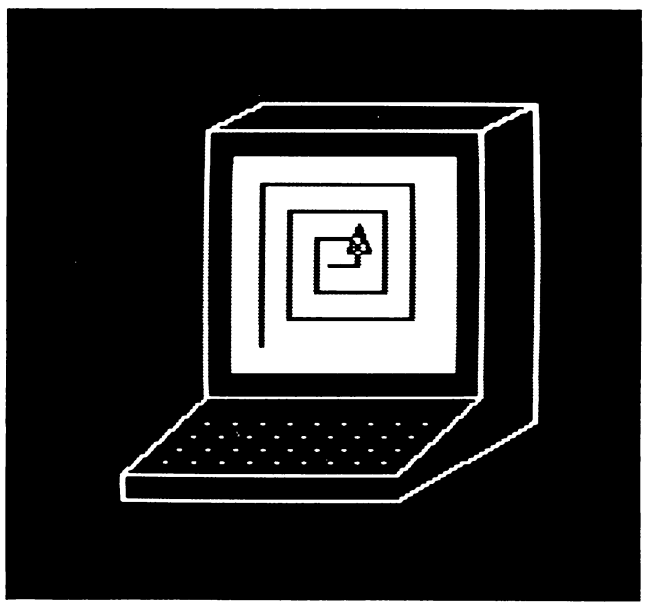


Figure 5-5.

Finally, you decide a message indicating the race has ended is necessary. You also want to give your final idea a more meaningful name, GAME. (Do this by changing the name of ALL in edit mode.) You make the following modifications.

```
TO WINNER
SETPOS [Ø 2Ø]
PR [THE TURTLE WINS!!!!!]
REPEAT 1Ø [HT WAIT 1Ø ST]
END
```

```
TO GAME
HT
COMPUTER
SCREEN
KEYS -5Ø -4Ø Ø Ø
PU SETPOS [-3Ø -1Ø] PD
MAZE 6Ø
ST
PU SETPOS [-25 -5] SETH Ø
MAZE2 5Ø
WINNER
END
```

Type GAME to observe the final diagram; Figure 5-6 illustrates how it should appear. Also save your proudly created project as VIDEO.GAME. Remember to ERPS before beginning the next section.

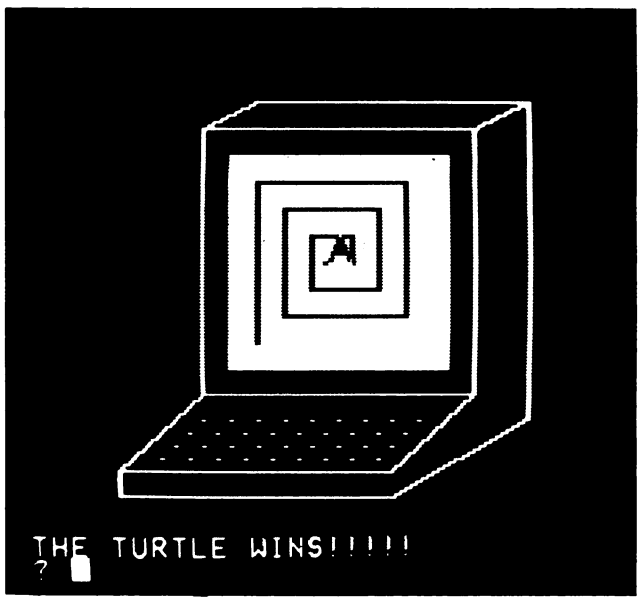


Figure 5-6.

Whole-To-Part Programming

Whole-to-part programming is an approach where the learner has a clearly defined idea or design in mind. The entire drawing is observed, studied, then divided into smaller parts. Each part is given a name, for which a procedure will be developed. Procedures are developed for each of the defined subparts. These procedures can be executed as they are

developed (to see the project grow) or all at one time (to see the project at once).

Often the ambitious programmer may conceive an idea with subparts that must be modified, eliminated, or substituted from the original plan. Occasionally, you will find that a mistake in a procedure creates a more pleasing display than the one originally intended, so it is incorporated into the final project. The following steps lead to successful whole-to-part programming.

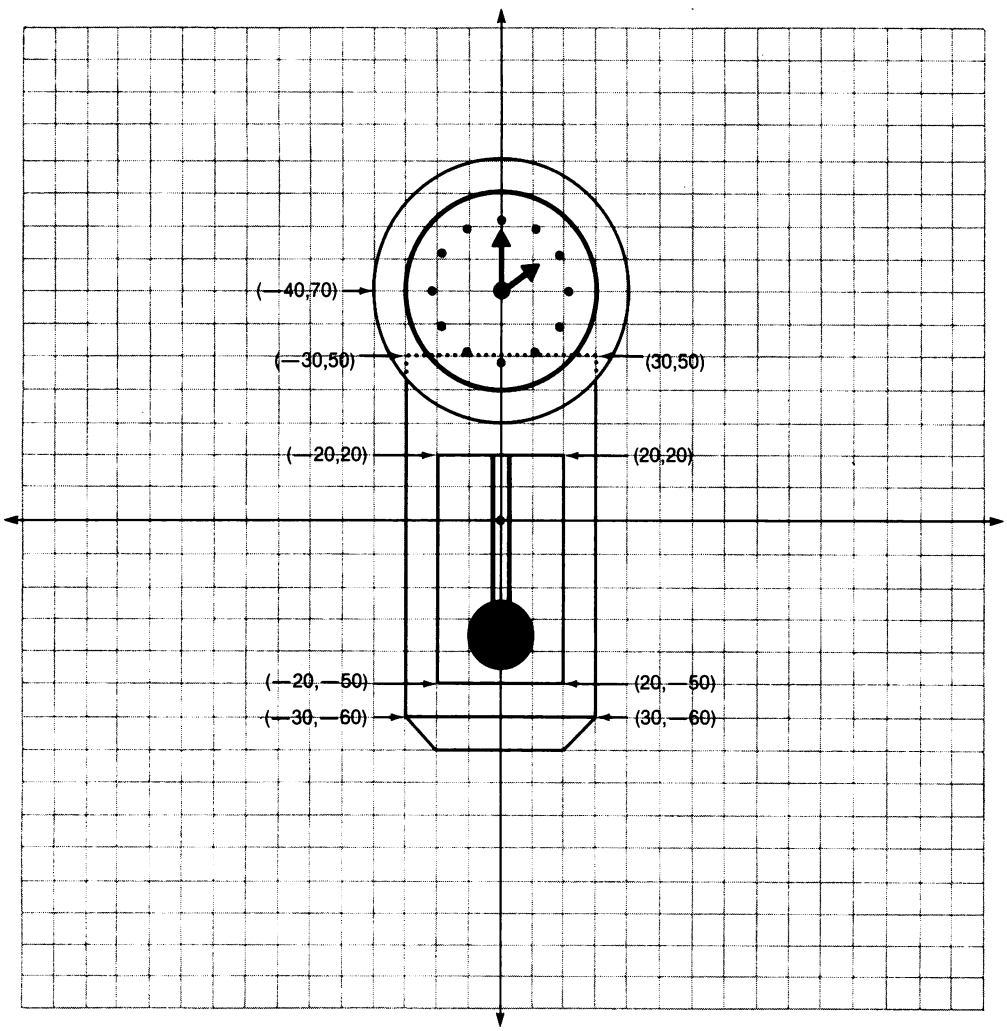


Figure 5-7.

Step 1. The first step in the whole-to-part approach to programming is to study the entire picture in detail. Examine the picture for structures that are similar and can be designed as general-purpose procedures, where different input variables distinguish one application of a procedure from another. Studying the picture in Figure 5-7, notice there are three rectan-

gles: the outer middle case, the pendulum, and the window for viewing the pendulum. There are also three circles: the bob for the pendulum, and two circles which form the upper clock case frame. The bottom of the case is a unique trapezoidal shape. The hands, initials, and dots for numbers on the clock are also distinct, non-repetitive drawings.

Step 2. The next step is to note any special features regarding the diagram. For example, let's suppose the clock is hanging on a wall that is wall-papered. In order to avoid the transparent interference of the clock and the wallpaper, the clock should be shaded rather than outlined.

Let's agree that the main clock case (the circular top, middle rectangle, and lower trapezoid) will be orange. The viewing window and dial (circular area where numbers and hands appear) will be white. The pendulum bob will be green. The pendulum, dots for numbers, initials, and hands will be black. Some highlighting in black will also give depth to the diagram.

Step 3. The third step is to determine a sequence by which the diagram will develop. In some cases this step is not important; in others it is.

In our example, the wallpaper would be displayed first. Then the clock will be drawn on top with the shading covering the wallpaper pattern appearing in those locations. Also, the longer rectangle needs to appear as if it is behind the circular part of the clock frame and dial, as indicated by the dotted line in Figure 5-7. In addition, it is convenient to shade the viewing window over the large rectangular part of the frame. The pendulum and bob would then be drawn on top of the window. The dial must be shaded white prior to the display of dots for numbers and hands. It is not important when the lower trapezoidal region is added, whether the hands are drawn before the number dots, or whether the window section is drawn before of after the upper circular region. The sequence determined by the authors is given below.

1. Shade large rectangle in orange
2. Shade window in white
3. Shade lower trapezoidal region in orange
4. Shade part of larger circle in orange
5. Continue with shading of circle, but in white
6. Display dots for numbers
7. Display hands
8. Draw initials
9. Shade pendulum
10. Shade pendulum bob
11. Draw lines to give the picture depth

To reflect this idea, procedures in CLOCK.SCENE are conceived. Enter the procedure defined below.

```
TO CLOCK.SCENE
CASE
```

```
FACE
TIME
HANDS
INITIALS
PENDULUM
EMPHASIS
END
```

Step 4. Now, it is time to develop these procedures. Let's first construct the procedure for coloring a rectangular region. There are several schemes which could be developed. The one adopted for this example is to color the rectangle by repetitively drawing horizontal lines from the lower boundary of the rectangle to its upper boundary. Figure 5-8 illustrates the points of importance in formulating this procedure.

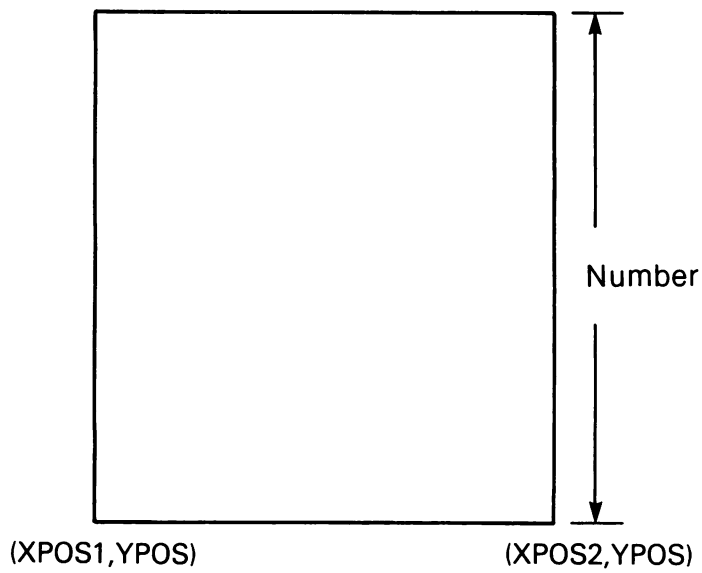


Figure 5-8.

The plan is to draw a horizontal line from XPOS1 to XPOS2 at row YPOS, then increase YPOS by 1 and continue to repeat this process. The number of repetitions is determined by the value of NUMBER.

RECTFILL defines the procedure that performs this process. Enter RECTFILL and test the procedure by entering the given input value. These are the actual values needed as one of the steps to create the clock. The value 110 is used since there are 110 (or 50 − (−60)) horizontal lines between the screen coordinates (30,50) and (30,−60). Refer to Figure 5-7 to verify these values. Similarly, 70 (or 20 − (−50)) horizontal lines are used to shade the window, and 50 (or 20 − (−30)) horizontal lines are used to shade the pendulum. Variable COLOR is used to control the changing of colors in the procedure.

```
TO RECTFILL :XPOS1 :XPOS2 :YPOS :COLOR :NUMBER
IF :NUMBER = Ø [STOP]
SETPC :COLOR
PU
SETPOS LIST :XPOS1 :YPOS
PD
SETPOS LIST :XPOS2 :YPOS
RECTFILL :XPOS1 :XPOS2 :YPOS + 1 :COLOR :NUMBER - 1
END

TO WEDGEFILL :XPOS1 :XPOS2 :YPOS :COLOR :NUMBER
IF :NUMBER = Ø [STOP]
SETPC :COLOR
PU SETPOS LIST :XPOS1 :YPOS PD
SETPOS LIST :XPOS2 :YPOS
WEDGEFILL :XPOS1 + 1 :XPOS2 - 1 :YPOS - 1 :COLOR :NUMBER - 1
END

TO CASE
RECTFILL -3Ø 3Ø -6Ø 4 11Ø
RECTFILL -2Ø 2Ø -5Ø 1 7Ø
WEDGEFILL -3Ø 3Ø -6Ø 4 1Ø
PU HOME PD
END

CLOCK.SCENE
```

The coloring of circles is achieved by repetitively drawing concentric circles, with each new circle having a radius of one less than the previous one. The CIRCLER procedure is used, but you must be careful to start the drawing at the correct point and at SETH 0. Coordinate (−40,70) in Figure 5-7 is used to begin coloring the larger circle.

The coloring needs only occur in orange ten times since the thickness of the circular frame is ten units. Then the same procedure can continue to draw concentric circles in white.

The COLORFILL procedure is given below. In this procedure, variable NUMBER controls the number of concentric circles to be drawn while COLOR sets the color. Enter the procedure and test it by using the given values, which are the actual ones needed for the diagram.

```
TO CIRCLEFILL :RADIUS :COLOR :NUMBER
SETPC :COLOR
IF :NUMBER = Ø [STOP]
CIRCLER :RADIUS
PU SETH 9Ø FD 1 SETH Ø PD
CIRCLEFILL :RADIUS - 1 :COLOR :NUMBER - 1
END

TO FACE
PU FD 7Ø SETH 27Ø FD 4Ø SETH Ø PD
CIRCLEFILL 4Ø 4 1Ø
```

```
CIRCLEFILL 30 1 27
END

FACE
```

The other procedures needed to complete the diagram are given below for you to study and enter. You might want to test each procedure as it is entered to observe the evolution of the clock.

```
TO TIME
SETPC 0
DOT [0 95]
DOT [12 89]
DOT [20 80]
DOT [25 70]
DOT [20 60]
DOT [12 51]
DOT [0 45]
DOT [-12 51]
DOT [-20 60]
DOT [-25 70]
DOT [-20 80]
DOT [-12 89]
END

TIME

TO HANDS
SETPC 0
PU SETPOS [0 70] PD
SETH 0 FD 20
SETH 150 FD 4 BK 4
SETH 210 FD 4
PU SETPOS [0 70] PD
SETH 60 FD 12
SETH 195 FD 4 BK 4
SETH 265 FD 4
END

HANDS

TO INITIALS
SETPC 0
PU SETPOS [-2 63] PD
SETPOS [-8 63]
SETPOS [-8 60]
SETPOS [-2 60]
SETPOS [-2 57]
SETPOS [-8 57]
PU SETPOS [-2 63] PD
SETPOS [8 63]
SETPOS [5 63]
```

```
SETPOS [5 57]
END

INITIALS

TO PENDULUM
RECTFILL -2 2 -3Ø Ø 5Ø
PU HOME SETH 18Ø FD 35 SETH 27Ø FD 1Ø SETH Ø PD
CIRCLEFILL 1Ø 2 1Ø
END

PENDULUM

TO EMPHASIS
SETPC Ø
PU SETPOS [-2Ø -5Ø] PD
SETPOS [-2Ø 2Ø]
SETPOS [2Ø 2Ø]
SETPOS [2Ø -5Ø]
SETPOS [-2Ø -5Ø]
PU SETPOS [-3Ø -6Ø] PD
SETPOS [3Ø -6Ø]
PU SETPOS [-27 -63] PD
SETPOS [27 -63]
PU SETPOS [-24 -66] PD
SETPOS [24 -66]
PU SETPOS [-42 7Ø] SETH Ø PD
CIRCLER 42
END

EMPHASIS
```

Step 5. The last step is to finally review all the tested procedures under one name. Study the sequence and input values of these commands and procedures used to create the diagram illustrated in Figure 5-7. Review the order of appearance of each clock part. Use the editor to modify CLOCK.SCENE, then execute the procedure to observe the final result, illustrated in Figure 5-9.

```
TO CLOCK.SCENE
HT
CASE
FACE
TIME
HANDS
INITIALS
PENDULUM
EMPHASIS
PU HOME PD
END

CLOCK.SCENE
SAVE "TIME.PIECE
```

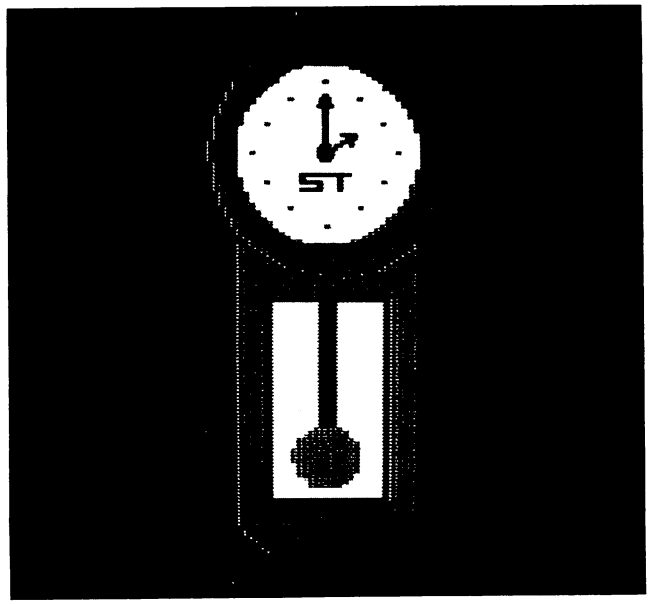


Figure 5-9.

Concluding Remarks

Congratulations! You have almost reached the end of this book. We hope you found the Logo activities a rewarding and motivating experience.

Logo is not a computer language one learns for the sake of learning a programming language. Instead, Logo is a vehicle by which problem solving and critical thinking take place. By mastering the fundamental concepts of Logo, you now have the opportunity to expand and disseminate its various applications in a learning environment.

First, if you are an educator, introduce or continue to use Logo in your classroom. Encourage your students to explore and elaborate on the activities you initiate. Get other teachers involved. Have the excitement of Logo permeate throughout your entire school district.

Secondly, if you are a parent or child, Logo provides a medium through which a family can learn together and share creative ideas at an academic level common to both groups. If you have read this book independently, introduce your family to Logo.

Finally, if you are a computer hobbyist, or one who simply enjoys working at a computer, you will want to suggest Logo to other hobbyists. Perhaps you will want to start a Logo group in your community where ideas and information are interchanged.

Learning with Logo is an ongoing process. Each interesting idea can serve as a catalyst for another. Your Logo experience should not end here. There are many challenges for you to accept in the Suggested Projects listed below. The Resource List also provides periodical articles, other books, and possible publications which will increase your understanding of Logo beyond *Apple Logo: Activities For Exploring Turtle Graphics.*

Enjoy Exploring Your Thoughts!

Suggested Projects

1. A circus wagon
2. A sunset at the beach
3. A floorplan
4. A spaceship
5. The solar system
6. A tennis court
7. A zoo
8. A sailboat regatta
9. A musical instrument
10. A campground
11. An original idea

Resource List

Books

Abelson, Harold: Apple Logo. BYTE/McGraw-Hill, New York, 1982

Bearden, Donna; Muller, James; Martin, Kathleen: The Turtle's Sourcebook. Reston Publishing Company, Reston, VA, 1983

Bitter, Gary G.; Watson, Nancy: Apple Logo Primer. Reston Publishing Company, Reston, VA, 1983

Conlan, Jim; Inman, Don: Sprites, A Turtle and TI Logo. Reston Publishing Company, Reston, VA, 1983

Papert, Seymour: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York, 1980

Papert, Seymour; diSessa, Andrea; Watt, Daniel; Weir, Sylvia: Final Report of the Brookline Logo Project. MIT Logo Group, Cambridge, MA, 1979

Ross, Peter: Introducing Logo for the Apple II Computer, Texas Instruments 99/4a, and Tandy Color Computer. Addison-Wesley Publishing Co., Reading, MA, 1983

Sharp, Pamela: Turtlesteps. Brady Communications Company, Inc., Bowie, MD 1984

Taylor, Robert P: The Computer in the School: Tutor, Tool, Tutee. Teacher's College Press, New York, 1980

Thornburg, David D: Discovering Apple Logo: An Invitation to the Art and Pattern of Nature. Addison-Wesley Publishing Co., Reading, MA, 1983

Thornburg, David D: Picture This! Pilot Turtle Geometry, An Introduction to Computer Graphics For Kids of All Ages. Addison-Wesley Publishing Co., Reading, MA, 1982

Watt, Daniel: Learning With Logo. BYTE/McGraw-Hill, New York, 1982

_____: Apple Logo in the Classroom. MECC Distribution Center, St. Paul, MN, 1983

_____: Cyberlogo Turtle: A First Step In Computer Literacy. Cybertronics International Inc., Reston Publishing Company, Reston, VA, 1983

Periodical Articles

Abbott, Hilton. "Three-Key Logo." Classroom Computer News, May-June 1983

Abelson, Harold: "A Beginner's Guide To Logo." BYTE, August 1982

Bearden, Donna: "The Voice Of The Turtle." Softalk, October-November-December 1983

Bearden, Donna; Muller, Jim: "Logo Talks Back," InCider, September 1983

Bearden, Donna; Muller, Jim: "Turtle Graphics: On and Off the Screen." Classroom Computer News, April 1983

Billstein, Rick: "Learning Logo and Liking It." Computing Teacher, November 1982

Bitter, Gary G: "The Road To Computer Literacy" (a five-part series). Electronic Learning, September-October-November 1982, January-February 1983

Bowen, Barbara: "Using Logo in the Classroom: The Project Approach." Teaching and Computers, Spring 1983

Carter, Richard: "Logo and the Great Debate." Microcomputing, September 1981

Carter, Ricky: "The Beginner's Guide to Logo." Classroom Computer News, April 1983

Cron, Mary: "Trouble in Logoland." Teaching, Learning, Computing, December 1983

Faflick, Phillip: "Teaching the Turtle New Tricks." Time, October 11, 1982

Harvey, Brian: "Why Logo?" BYTE, August 1982

Kildall, Gary; Thornburg, David: "Digital Research's DR Logo." BYTE, June 1983

Lawler, Robert: "Designing Computer-Based Microworlds." BYTE, August 1982

Lough, Tom: "Exploring New Horizons with Logo: How One School is Charting the Course. Part 2." Electronic Learning, April 1983

Lough, Tom: Logo: Discovery Learning with the Classroom's Newest Pet." Electronic Learning, March 1983

Lough, Tom: "WATERCROSS: A Logo Exploration." Computing Teacher, November 1982

Lough, Tom; Tipps, Steve: "Logo Notebook." Teaching and Computers, September 1983

Lough, Tom; Tipps, Steve: "Is There Logo After Turtle Graphics?" Classroom Computer News, April 1983

Markuson, Carolyn; Tobias, Joyce; Lough, Tom: "Logo Fever: The Computer Language Every School is Catching." Instructor, January 1983

Moore, Mary Jo: "The Art of Teaching Logo." Classroom Computer News, April 1983

Muller, Jim: "Logo: the Voice of the Turtle." Softalk, December 1982

Muller, Jim: "The Friendly Computer Language." Creative Computing, October 1982

Nelson, Harold: "Logo: Not Just For Kids." Microcomputing, March 1982

Overall, Theresa et al: "Learning with Logo at the Lamplighter School." Microcomputing, September 1981

Papert, Seymour: "New Cultures from New Technologies." BYTE, September 1980

Piele, Donald T: "Beyond Turtle Graphics." Creative Computing, March 1983

Riordon, Tim: "Creating a Logo Environment." Computing Teacher, November 1982

Rosser, Phyllis: "Computers in the Classroom: How They May Change the Way Children Learn." Teachers and Writers Magazine, March-April 1983

Rousseau, Joseph; Smith, Stephen: "Wither Goes The Turtle?" Microcomputing, September 1981

Soloman, Cynthia: "Introducing Logo to Children." BYTE, August 1982

Stone, Greg: "Of Cats and Turtles in Mathland." InCider, June 1983

Stone, Greg: "Solving Problems with Logo." InCider, April 1983

Upitis, Rena: "Logo and the Primary-Junior Pupil: One Student's First Encounter." Computing Teacher, November 1982

Watt, Daniel: "A Comparison of the Problem Solving Styles of Two Children Learning Logo." Proceedings of the National Computing Conference, 1979

Watt, Daniel: "Logo in the Schools." BYTE, August 1982

Watt, Daniel: Teaching Turtles: Logo as an Environment for Learning." Popular Computing, July 1982

Watt, Molly: "What is Logo?" Creative Computing, October 1982

Weinreb, William: "Problem Solving with Logo." BYTE, November 1982

Weir, Sylvia: "Logo and the Exceptional Child." Microcomputing, September 1981

Williams, Greg: "Logo for the Apple II, the TI-99/4A, and the TRS-80 Color Computer." BYTE, August 1982

_____: "Input/Output: Ideas from the Classroom." Classroom Computer News, April 1983

_____. "The Logo Journal, News and Views of the Logo Community—a Collection of Articles." BYTE, August 1982

Subscriptions

TurtleTalk (four issues per year)
TurtleTalk
955 Greenwood Blvd.
Issaquah, WA 98027

Turtle News (six issues per year)
Young People's Logo Association
P.O. Box 855067
Richardson, TX 75085

Answers To Turtle Testers

Chapter 1

Answer to Activity 1.1 Tester (page 6)

```
CS
HT
FD 12Ø
CS
ST
BK 75
END
```

Answer to Activity 1.2A Tester (page 11)

```
FD 6Ø
RT 9Ø
FD 4Ø
RT 9Ø
FD 6Ø
RT 9Ø
FD 4Ø
RT 9Ø
END
```

Answer to Activity 1.2B Tester (page 11)

```
FD 4Ø
LT 9Ø
FD 8Ø
LT 9Ø
FD 4Ø
LT 9Ø
FD 8Ø
LT 9Ø
```

Answer to Activity 1.3 Tester (page 16)

```
BK 5Ø
FD 1ØØ
RT 9Ø
FD 5Ø
RT 9Ø
FD 45
RT 45
FD 1Ø
SETH 27Ø
FD 43
BK 43
SETH 135
FD 1Ø
RT 45
FD 45
RT 9Ø
FD 5Ø
RT 9Ø
```

Answer to Activity 1.4 Tester (page 21)

```
PU
FD 3Ø
PD
REPEAT 6 [FD 4Ø LT 36Ø/6]
REPEAT 6 [FD 4Ø RT 36Ø/6]
RT 12Ø
FD 4Ø
RT 12Ø
FD 4Ø
SETH Ø
BK 4Ø
REPEAT 6 [FD 4Ø LT 6Ø]
REPEAT 6 [FD 4Ø RT 6Ø]
PU HOME PD
```

Chapter 2

Answer to Activity 2.1 Tester (page 41)

```
TO PENTAGON
REPEAT 5 [FD 4Ø RT 36Ø/5]
END

TO TESTER2.1
REPEAT 6 [PENTAGON RT 36Ø/6]
END

TESTER 2.1
```

Answer To Activity 2.2 Tester (page 50)

```
TO TRIANGLE
REPEAT 3 [FD 35 RT 36Ø/3]
END

TO DECAGON
REPEAT 1Ø [FD 35 RT 36Ø/1Ø]
END

TO NONAGON
REPEAT 9 [FD 35 RT 36Ø/9]
END

TO OCTAGON
REPEAT 8 [FD 35 RT 36Ø/8]
END

TO HEPTAGON
REPEAT 7 [FD 35 RT 36Ø/7]
END

TO HEXAGON
REPEAT 6 [FD 35 RT 36Ø/6]
END

TO PENTAGON
REPEAT 5 [FD 35 RT 36Ø/5]
END
```

Chapter 3

Answer to Activity 3.1 Tester (page 94)

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 9Ø]
END

TO SQUARES.2
SQUARE 1Ø
SQUARE 3Ø
SQUARE 5Ø
SQUARE 7Ø
HT
END

TO SQDESIGN.2 :MANY
REPEAT :MANY [SQUARE .2 RT 36Ø/:MANY]
END
```

Answer to Activity 3.6 Tester (page 138)

```
TO CIRCLES :RADIUS
REPEAT 5 [CIRCLER :RADIUS RT 72]
END

TO MANY.CIRCLES :RADIUS :INCREMENT
FENCE
CIRCLES :RADIUS
MANY.CIRCLES :RADIUS + :INCREMENT :INCREMENT
END

MANY.CIRCLES 30 10
WRAP
```

Answer to Activity 3.7 Tester (page 151)

```
TO FILL :SIDE :INCREMENT :MANY
IF :SIDE < :INCREMENT [STOP]
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :INCREMENT :INCREMENT :MANY
END

TO COLORFILL :SIDE :INCREMENT :MANY :COLOR
SETPC :COLOR
FILL :SIDE :INCREMENT :MANY
END

TO ARRANGE
FULLSCREEN
MOVE 270 40
MOVE 0 20
COLORFILL 50 .5 5 3
MOVE 90 120
COLORFILL 25 .4 8 4
MOVE 180 100
COLORFILL 20 .4 6 5
MOVE 270 120
COLORFILL 25 .3 10 2
PU HOME PD
END

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH 0 PD
END

ARRANGE
```

Answer to Activity 3.8 Tester (page 158)

```
TO GYROTRI :SIDE
HT
RT 30
TRIANGLE :SIDE
END
```

```
TO TRIANGLE :SIDE
IF :SIDE < 4 [STOP]
REPEAT 3 [FD :SIDE RT 12Ø]
FD :SIDE/2 RT 6Ø
TRIANGLE :SIDE/2
END
```

Chapter 4

Answer to Activity 4.1 Tester (page 175)

```
TO LONG :YPOSITION
PU SETX Ø
SETY :YPOSITION
PD FD :YPOSITION * -2
END

TO LAT :XPOSITION
PU
SETX :XPOSITION
SETY Ø
SETH 9Ø
PD FD :XPOSITION * -2
END

TO NEGATIVE :SIDE
PD
SETH 9Ø
FD :SIDE * 2
END

TO POSITIVE :SIDE
PD REPEAT 4 [FD :SIDE BK :SIDE RT 9Ø]
END

TO RBRACKET :SIDE :XPOSITION :YPOSITION :ANGLE
MOVE :XPOSITION :YPOSITION :ANGLE
FD :SIDE
SETH Ø
FD :SIDE * 2
SETH 27Ø
FD :SIDE
END

TO LBRACKET :SIDE :XPOSITION :YPOSITION :ANGLE
MOVE :XPOSITION :YPOSITION :ANGLE
FD :SIDE
SETH Ø
FD :SIDE * 2
SETH 9Ø
FD :SIDE
END
```

```
TO AXIS :XPOSITION :YPOSITION
LONG :YPOSITION
LAT :XPOSITION
END

TO GRID.LABELS
AXIS -139 -12Ø
LBRACKET 1Ø 3Ø 4Ø 27Ø
RBRACKET 1Ø 8Ø 4Ø 9Ø
MOVE 4Ø 5Ø Ø
POSITIVE 1Ø
MOVE 65 5Ø Ø
POSITIVE 1Ø
LBRACKET 1Ø 3Ø -4Ø 27Ø
RBRACKET 1Ø 8Ø -4Ø 9Ø
MOVE 4Ø -3Ø Ø
POSITIVE 1Ø
MOVE 65 -3Ø Ø
NEGATIVE 1Ø
LBRACKET 1Ø -8Ø 4Ø 27Ø
RBRACKET 1Ø -3Ø 4Ø 9Ø
MOVE -8Ø 5Ø Ø
NEGATIVE 1Ø
MOVE -4Ø 5Ø Ø
POSITIVE 1Ø
LBRACKET 1Ø -8Ø -4Ø 27Ø
RBRACKET 1Ø -2Ø -4Ø 9Ø
MOVE -8Ø -3Ø Ø
NEGATIVE 1Ø
MOVE -4Ø -3Ø Ø
NEGATIVE 1Ø
HT
END

TO MOVE :XPOSITION :YPOSITION :ANGLE
PU SETPOS LIST :XPOSITION :YPOSITION
SETH :ANGLE
PD
END

GRID.LABELS
```

Answer to Activity 4.2 Tester (page 186)

```
TO ROW :XPOS :YPOS :TALLYCOLUMN
DOT LIST :XPOS :YPOS
IF :TALLYCOLUMN = 9 [STOP]
ROW :XPOS + 1Ø :YPOS :TALLYCOLUMN + 1
END

TO STATES :XPOS :YPOS :TALLYCOLUMN :TALLYROW
SETPC 4
ROW :XPOS :YPOS :TALLYCOLUMN
```

```
IF :TALLYROW = 4 [STOP]
STATES :XPOS :YPOS - 1Ø :TALLYCOLUMN :TALLYROW + 1
END

TO COLORFILL :SIDE :INCREMENT :MANY :COLOR
SETPC :COLOR
FILL :SIDE :INCREMENT :MANY
END

TO FILL :SIDE :INCREMENT :MANY
IF :SIDE < :INCREMENT [STOP]
REPEAT :MANY [FD :SIDE RT 36Ø / :MANY]
FILL :SIDE - :INCREMENT :INCREMENT :MANY
END

TO STARS
PU SETPOS [-98 9Ø]
PD SETH 9Ø
COLORFILL 6Ø 1 4 5
PU FD 4Ø
PD SETH 9Ø
COLORFILL 6Ø 1 4 5
STATES -9Ø 8Ø Ø Ø
END

TO STRIPES
SETPC 1
PU SETPOS [-99 9Ø]
PD SETPOS [13Ø 9Ø]
SETPOS [13Ø 8Ø]
SETPOS [-99 8Ø]
PU SETPOS [-99 7Ø]
SETPC 3
PD SETPOS [13Ø 7Ø]
SETPOS [13Ø 6Ø]
SETPOS [-99 6Ø]
PU SETPOS [-99 5Ø]
PD SETPC 1
SETPOS [13Ø 5Ø]
SETPOS [13Ø 4Ø]
SETPOS [-99 4Ø]
PU SETPOS [-99 3Ø]
SETPOS [-99 3Ø]
PD SETPC 3
SETPOS [13Ø 3Ø]
SETPOS [13Ø 2Ø]
SETPOS [-99 2Ø]
PU SETPOS [-99 1Ø]
PD SETPC 1
SETPOS [13Ø 1Ø]
SETPOS [13Ø Ø]
SETPOS [-99 Ø]
```

```
PU SETPOS [-99 -1Ø]
PD SETPC 3
SETPOS [13Ø -1Ø]
SETPOS [13Ø -2Ø]
SETPOS [-99 -2Ø]
PU SETPOS [-99 -3Ø]
PD SETPC 1
SETPOS [13Ø -3Ø]
SETPOS [13Ø -4Ø]
SETPOS [-99 -4Ø]
PU SETPOS [13Ø -4Ø]
PD SETPOS [13Ø 9Ø]
END

TO POST
FULLSCREEN
PU SETPOS [-11Ø -115]
PD SETPOS [-11Ø 9Ø]
SETPOS [-1ØØ 9Ø]
RT 9Ø
BK 5
CIRCLEL 1Ø
FD 5
SETPOS [-1ØØ -115]
SETPOS [-11Ø -115]
END

TO FLAG
POST
STRIPES
STARS
HT
END

FLAG
```

Answer to Activity 4.3 Tester (page 193)

```
TO BOW
SETPC 1
RT 45
REPEAT 3 [FD 8 RT 12Ø]
LT 9Ø
REPEAT 3 [FD 8 LT 12Ø]
HT
END

TO FRAME :XPOS :YPOS :C1
PU
SETPOS SE :XPOS :YPOS
PD SETPC :COLORFRAME
SETPOS SE :XPOS :YPOS + 5Ø
SETPOS SE :XPOS - 2Ø :YPOS + 3Ø
```

```
SETPOS SE :XPOS :YPOS
SETPOS SE :XPOS + 20 :YPOS + 30
SETPOS SE :XPOS :YPOS + 50
SETPOS SE :XPOS + 20 :YPOS + 30
SETPOS SE :XPOS - 20 :YPOS + 30
PU
SETPOS SE :XPOS :YPOS
SETPC 1
PD
END

TO KITE :XPOS :YPOS :COLORFRAME :COLORTAIL
FRAME :XPOS :YPOS :COLORFRAME
TAIL :XPOS :YPOS :COLORTAIL
PU HOME PD
END

TO MANY.KITES
FULLSCREEN
KITE 0 0 3 1
KITE -50 10 4 2
KITE 50 -10 1 5
END

TO TAIL :XPOS :YPOS :COLORTAIL
SETPC :COLORTAIL
SETH TOWARDS SE :XPOS - 10 :YPOS - 10
FD 16
SETH TOWARDS SE :XPOS - 16 :YPOS - 20
FD 8
SETH TOWARDS SE :XPOS - 14 :YPOS - 20
FD 8
BOW
SETPC :COLORTAIL
SETH TOWARDS SE :XPOS - 7 :YPOS - 27
FD 16
SETH TOWARDS SE :XPOS - 24 :YPOS - 49
FD 16
BK 10
BOW
END

MANY.KITES
```

Answer to Activity 4.4 Tester (page 199)

```
TO STAR.TESS
ACROSS 40 -115 -120 8
HT
END

TO SIDE :DISTANCE
RT 45 FD :DISTANCE
```

```
LT 45 FD :DISTANCE
LT 135 FD :DISTANCE
LT 45 FD :DISTANCE
RT 180
END

TO VARICUBE :LENGTH
BOX :LENGTH
SIDE :LENGTH
RT 90
FD :LENGTH
LT 90
SIDE :LENGTH
LT 90
FD :LENGTH
RT 135
FD :LENGTH
LT 45
BOX :LENGTH
RT 45
BK :LENGTH
LT 45
END

TO COLOR
SETPC (RANDOM 5) + 1
END

TO BOX :LENGTH
REPEAT 4 [FD :LENGTH RT 90]
END

TO ACROSS :LENGTH :XPLACE :YPLACE :DISTANCE
IF :XPLACE > 139 - :LENGTH [STOP]
VROW :LENGTH :XPLACE :YPLACE :DISTANCE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE :DISTANCE
END

TO STAR :DISTANCE :MANY
COLOR
REPEAT :MANY [VARICUBE :DISTANCE RT 360/:MANY]
PR POS
END

TO VROW :LENGTH :XPLACE :YPLACE :DISTANCE
FULLSCREEN
PU SETPOS LIST :XPLACE :YPLACE PD
MAKE "FIRSTPOS POS
MAKE "NUMBER (2 * :YPLACE)/:LENGTH
IF :NUMBER < 0 [MAKE "NUMBER :NUMBER * -1]
MAKE "MANY 8
```

```
REPEAT :NUMBER [STAR :DISTANCE :MANY MOVE :LENGTH]
PU SETPOS :FIRSTPOS PD
END

TO MOVE :LENGTH
PU FD :LENGTH PD
END
```

STAR.TESS

Index

angle, 7, 8
arc, 28, 30
ARCL, 29–31, 33, 38
ARCR, 29–31, 33, 38
arithmetic operations, 159

BACK (BK), 2–6, 32
brackets [ ], 17, 18

CATALOG, 43, 50, 51, 53
circle, 22, 23, 26
CIRCLEL, 24–27, 33, 38
CIRCLER, 23–27, 33, 38
CLEAN, 14–16, 33
CLEARSCREEN (CS), 2–6, 32
CLEARTEXT, 52, 54, 55, 122, 159
CO, 120, 122, 160
cold boot, 1
color chart, 98
constants, 90–91
CONTROL (CTRL) key, 3, 68
correcting typing errors, 2
CTRL-A, 57–60
CTRL-B, 57–60
CTRL-C, 56, 57, 61
CTRL-D, 57–60
CTRL-E, 57, 59
CTRL-G, 37, 58, 120, 122, 195
CTRL-K, 57–60, 75, 182, 184
CTRL-L, (FULLSCREEN), 3–6, 32
CTRL-N, 56, 58, 59, 60
CTRL-O, 56, 58, 59, 60
CTRL-P, 56, 58, 59, 60
CTRL-S (SPLITSCREEN), 3–6, 32
CTRL-T (TEXTSCREEN), 3–6, 32
CTRL-W, 54, 120, 122, 160
CTRL-Y, 57–60, 75, 181, 182, 184
CTRL-Z, 120, 122, 160
cursor, 2

definition mode, 36–41, 74, 75, 77, 78
DOT, 177, 184

EDIT, 55, 56, 60
edit mode, 75, 77–78
ER, 59, 61, 161
ERASEFILE, 53–55
ERPS, 46, 47, 49, 52, 54
exterior angle, 19, 20

FENCE, 127, 130, 135, 136, 159
FORWARD (FD), 2–6, 32
FULLSCREEN (CTRL-L), 3–6, 32, 189

global variable, 116, 117, 142, 150, 160

HEADING, 6–11
HIDETURTLE (HT), 2–6, 32
HOME, 10, 11, 15, 32

IF (conditional expression), 140–145, 160
initialize diskette (INIT), 35, 36, 38
input, 2

LIST, 172, 173, 189, 202
LOAD, 51, 53, 54
local variables, 116, 117, 142, 160
LT, 7–11, 32

MAKE, 113–117, 124, 142, 160

ordered pair (X,Y), 168, 172, 201–202

part-to-whole programming, 210–216
PEN, 110–111, 116
PENDOWN (PD), 13–16, 32
PENERASE (PE), 13–16, 33, 110–112, 116
PENREVERSE (PX), 110–111, 116, 158
PENUP (PU), 13–16, 32
pixel, 184
PO, 52, 54
polygon, 19
PONS, 133, 136, 159
POPS, 52, 54
POTS, 51–55, 161
predicates and functions ([,], =), 140–145, 160
primitives, 36
PRINT (PR), 9, 32, 102, 105, 106, 108, 141, 160
PR HEADING, 9–11, 32
procedures, 36, 40, 46, 47
prompt, 2
PR PEN, 110, 116, 160
PR POSITION (PR POS), 168–170, 172, 180, 203
PR TOWARDS, 177, 178, 180, 184, 203
PR XCOR, 170, 173, 202
PR YCOR, 170, 173, 202–203
PX (PENREVERSE), 110, 111, 116, 158

RANDOM, 115, 117, 124, 160
recursion (complex), 127–128, 152–155
recursion (simple), 119–126, 140–145, 152, 154–155
REPEAT, 18–21, 33
RT, 7–11, 32

SAVE, 46, 47, 49, 53, 95
screen (Cartesian) coordinates, 167–168, 172, 191, 201
SENTENCE (SE), 172, 173
SETBG, 100–106, 158
SETH, 9–11, 32
SETH TOWARDS, 177, 179, 184, 203
SETPC, 96–98, 103–107, 158
SETPEN, 158
SETPOS, 170, 171, 173, 180, 191, 202
SETX, 169, 173, 202
SETX, 169, 173, 202
SHOWTURTLE (ST), 2–6, 32
SPLITSCREEN (CTRL-S), 3–6, 32, 188
STARTUP file, 1, 26, 36, 38, 43
STOP, 141–145, 155

tessellations, 194–199, 202
TO (definition mode), 36–41
Total Turtle Trip Theorem, 19, 30
TEXTSCREEN (CTRL-T), 3–6, 32
toplevel, 73, 74, 77–78
TYPE, 120

variables, 85–92, 96, 99, 106, 107, 160

WAIT, 154, 160
warm boot, 35, 36
whole-to-part programming, 216–223
WINDOW, 128, 135–136
workspace, 47, 50, 53, 54, 59
WRAP, 128, 130, 135–136, 159

Free Subscription!

Welcome to APPLE CHANNEL™—a brand-new information service for APPLE users.

What you'll get is a *personal* report, updated quarterly, covering the latest Apple books, games and software available from Brady. Current, informative and easy-to-use, APPLE CHANNEL™ also offers you valuable *advance* information on forthcoming new products.

To begin your *FREE* subscription, complete and mail this card today! No postage (or purchase) required.

(Please print or type)

Name _____

Address _____

City _____ State _____ Zip _____

How do you usually find out about new books and software?

☐ Magazine Advertisements ☐ Mail at Work or Home

☐ Book/Computer Store Browsing ☐ Friend's Recommendation

☐ Other (please explain) _____

What areas are you most interested in?

☐ Business Applications ☐ Education

☐ Home Applications ☐ Games

☐ Other (please explain) _____

Today's Date _____ Signature _____

BRADY BOOKWARE™
Robert J. Brady Co. • Bowie, MD 20715
A Prentice-Hall Company D312X-8

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1976 BOWIE, MD.

Postage will be paid by addressee

Brady Communications Co., Inc.
A Prentice-Hall Company
Bowie, Maryland 20715

Parents, Teachers! Here's An Exciting And Creative New Way To Explore Turtle Graphics Together With Your Children and Students!

Acclaim from prepublication reviewers:

". . . an excellent resource for the teacher who's suddenly faced with the necessity of teaching Logo without benefit of formal coursework. . ."

". . . its greatest strength is its 'hands-on' approach to teaching Logo . . . no other book competes with this one!"

". . . the activities are extremely well thought out and properly ordered . . ."

APPLE LOGO: ACTIVITIES FOR EXPLORING TURTLE GRAPHICS

Harold J. Bailey, Kathleen M. Brautigam, and Trudy H. Doran

Now—a "hands-on" activities guide that offers children an exciting and creative approach to exploring turtle graphics! This guide includes a series of "hands-on" activities that begin with basic turtle commands and concepts . . . then expand to more sophisticated graphics design. What's more, this book goes beyond the "discovery-based" approach to learning Logo—inside this book you'll undergo the "guided discovery" approach that gives youngsters the ability to grasp powerful ideas through systematic planning and problem-solving techniques!

Divided into five sequentially designed chapters, *Apple Logo: Activities for Exploring Turtle Graphics* shows users how to control movement of the "turtle" through simple Logo commands . . . how to combine these commands to form "procedures". . . how to use color . . . how to use coordinates for effective graphics design . . . and how to put it all together to produce complex screen displays! Each activity presented within a chapter is divided into five concise "turtle" sections as well, offering users immediate and practical feedback from the initially given Logo command to the final activity evaluation!

CONTENTS

EXPLORING TURTLE TERRITORIES . . . Turtle Moves/Turtle Pivots/Turtle Toggles/Turtle Repetitions/Turtle Circles/Turtle Arcs/Reference Point One
PRESERVING TURTLE TRACKS . . . Turtle Procedure/Turtle Files/Turtle Inventory/Turtle Editing/Turtle Controls/Turtle Maintenance/Reference Point Two
EXTENDING TURTLE TRIPS . . . Turtle Input/Turtle Hues/Turtle Background/Turtle Variables/Turtle Iterations/Turtle Limits/Turtle Conditions/Turtle Tricks/Reference Point Three
COORDINATING TURTLE TRACES . . . Turtle Positions/Turtle Dots/Turtle Placements/Turtle Tessellations/Reference Point Four
CREATING TURTLE TEMPTATIONS

ISBN 0-89303-312-X